UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA
DEPARTAMENTO DE COMUNICAÇÕES

UNICAMP

Novas investigações para implementação de um ambiente de trabalho para Processamento Digital de Sinais

Tese submetida à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas, Departamento de Comunicações, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica.

Autor

António Maria Fançony Gaspar
Engenheiro Eletricista - UFES - 1992

Orientador

Prof. Dr. Yuzo Iano

Campinas, 21 de fevereiro de 1995.

Resumo

Este trabalho tem como objetivo apresentar a implementação da versão inicial do Ambiente de Trabalho para o Sistema Digital de Visualização e Processamento de Imagens - SDVI. Através da ferramenta Devguide - 1.1, desenvolveu-se dentro do OpenWindows, sistema operacional UNIX, o ambiente de trabalho que veio a denominar-se SDVI. Apresentamos ao longo deste trabalho a dimensão do projeto do sistema SDVI, para isso são também referenciadas várias publicações de nossa autoria com propostas de implementação em Hardware e Software da estrutura do sistema. Apresentamos a técnica de implementação do ambiente criado, as várias partes em que se compõe, o Software gerado, funções já existentes do grupo de PDI/DECOM adicionadas bem como pacotes de software públicos anexados. Por último apresentamos programas de conversão de freqüências de sinais em componentes RGB usando para isso a teoria de amostragem, sua Implementação, Simulação e Resultados.

Agradecimentos

Ao Prof. Dr. Yuzo Iano, pela amizade, apoio e orientação eficaz deste trabalho.

Aos Profs. Fábio Violaro, Rege Scarabucci, Hélio Waldman, Michel Yacoub, José G. Chiquito, João B. Yabu-uti e ao Prof. Leonardo S. Mendes pela amizade e apoio acadêmico.

Aos Profs. António Manuel, Calazans, Juçara e Edson Cardoso da Universidade Federal do Espírito Santo pelo apoio e amizade.

Aos colegas Ayres Mardem, Rossini, Carlos Pingarilho, Ademir, Sureros, Rodrigo, Alexandre, Jogurta, Marcelo Segatto e todos outros do curso de mestrado pela amizade e incentivo.

Aos amigos Joselito S. Cruz, Rober Marconi, Luiz, Fábio e Rogério pela amizade, companheirismo e apoio.

Aos funcionários da FEE pelo auxílio e em particular ao Jean e ao Mário pelo prestativo apoio.

A CAPES e ao Governo Brasileiro pelo apoio financeiro e oportunidade oferecida.

A todos aqueles que contribuíram para essa empreitada.

Finalmente,

Aos meus queridos Pais, Irmãos e a Deus.

Dedico este trabalho a minha esposa Ana Lúcia Bottecchia, ao meu filho Vinícius e aos meus pais e irmãos pelo prestativo apoio e íntima consideração.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1 Considerações iniciais

O processamento e a transmissão digital de imagens constituem um campo que cresceu consideravelmente na última década, devido às várias vantagens já conhecidas, inerentes aos sistemas digitalizados com relação aos sistemas analógicos [14] [15].

As vantagens da digitalização incluem a flexibilidade de processamento, facilidade de armazenamento, facilidade de recuperação, transmissão com regeneração digital do sinal, transmissão com precisão controlada, transmissão com registros precisos, facilidades para uso de códigos secretos, assim como compatibilidade com redes e computadores digitais [15].

O processamento digital de vídeo envolve a manipulação extremamente alta de dados de imagens ou sequências; isso inclui a versão original e processada.

Inúmeros centros de pesquisa no exterior têm estudado o processamento digital de vídeo visando às mais variadas aplicações [4], [5], [8], [24]. Dentre essas, pode-se citar a TV comercial em cores, o videofone, o fac-símile, imagens paradas, teleconferências, reconhecimento de padrões, enriquecimento de imagens, como por exemplo as recebidas por satélite, aplicadas na mineração, meteorologia, estudos estatísticos de poluição, desmatamento, áreas cultivadas e de forma geral, imagens utilizadas para análise e interpretação. Nessas aplicações não se procura apenas a digitalização do sinal analógico mas, principalmente a redução da taxa de bits.

Uma das limitações da transmissão analógica ou digital da informação de vídeo é a grande capacidade de canal requerida para se recuperar uma imagem de alto padrão de qualidade[24].

Dessa forma, tem sido considerável o investimento que vários centros de pesquisa dispõem para a criação de sistemas computarizados com hardware e software dedicados, construídos com objetivos de Digitalização, Processamento, Visualização e Transmissão de sinais de vídeo [4], [5], [8], [20], [22] e [25].

As dificuldades da construção de hardware para processamento em tempo real são o alto custo e o tempo requerido para se reproduzir o modelo proposto. Assim, o primeiro passo na investigação de novas técnicas, baseia-se na simulação de pequenos segmentos de sinais de vídeo armazenados. Embora esta aproximação tenha grande flexibilidade, é severamente limitada pelo tempo de armazenamento e processamento necessários. Por exemplo, 1 seg de um sinal de vídeo NTSC amostrado a uma freqüência de 4fsc requer aproximadamente 11MBytes de

memória e acima de 1 hr de processamento em CPU, tempo para um algorítmo complexo em um computador VAX 8600 [5]. Como vimos, devido ao número elevado de operações, faz-se necessário a existência de hardware dedicado, apresentando características programáveis dentro dos processadores, para se obter um sistema compacto de custo razoável. Existem propostas para sistemas mais generalizados como é o Princeton Engine [1] e o Processador de Sinal de Vídeo [2] onde múltiplos processadores operam massivamente em configuração paralela. Contudo, estes sistemas são grandes e em geral projetos inteiros são redefinidos para atualização.

O INRS-Télécommunications (CANADA) construiu o Sistema de Vídeo em Tempo-Real (RVS) [5]. A motivação inicial para a construção do sistema foi o desenvolvimento de novos algoritmos para a codificação e decodificação de sinais de vídeo para o sistema em cores NTSC, usando dois filtros bi-dimencionais não separáveis do tipo FIR no transmissor - pré-filtro de luminância e crominância antes da modulação - e receptor - filtro de banda passante para extração de crominância . Contudo, o RVS, é uma plataforma mais generalizada para vários tipos de pesquisa em processamento de vídeo em tempo real. Possui interfaces RGB e NTSC de entrada/saída que podem digitalizar nas freqüências de 13.5Mhz e 4fsc. Possui ainda versões digitais de circuitos analógicos de televisão convencionais (matrizes, modulador/demodulador). Excepto para as interfaces de entrada e saída o sistema é inteiramente digital. A arquitetura flexível permite a inserção de placas de circuitos com funções especializadas [5].

Em um contexto geral, a visualização de imagens permitindo exames e análises, requer hardware especializado acoplado a uma boa INTERFACE HOMEM-MÁQUINA.

O VIDS (Visual Display WorkStation do INRS-Telecommunications/Bell - Northen) foi construído baseado em dois conceitos, um "buffer" feito com semicondutores para segurar um número suficiente de sequências de um certo tamanho e operações de imagens dirigidas por software. O "buffer" permite visualização em tempo real mesmo em alta velocidade, combinando a velocidade do circuito de memória com o número de pixels de saída em paralelo. Com uma taxa, "memory bandwidth", alcançando 160MBytes/s é possível a reprodução de quadros inteiros ou aquisição de todos os sinais incluindo HDTV. Todas as operações de visualização são conduzidas por software. Um computador pessoal provê uma plataforma de comandos da qual derivam as ordens. Essas ordens incluem carregamento de imagens/sequências, visualização de uma ou outra sequência, chaveamento entre duas sequências, redimensionamento de imagens, etc... Um processador residente na WorkStation, recebe todas essas ordens e cria os efeitos mostrados no monitor, reduzindo assim consideravelmente o hardware necessário [4].

Portanto, são várias [4], [5], [6], [25]...as pesquisas que nos mostram o grande interesse de investimento na formação de sistemas computarizados para o tratamento de sinais de vídeo. Conclui-se que o sistema deve ter interfaces de entrada que permitem a aquisição de sinais de vídeo digitalizados, memória rápida e suficiente dada a quantidade de dados a manipular, hardware dedicado (Processadores Gráficos, Processadores Digitais de Sinais,...) devido a necessidade de realização de milhões de operações por segundo e uma interface homem-máquina capaz de operacionalizar o sistema.

[1] D.Chin et al., "The Princeton Engine: A Real-Time Video System Simulator", IEEE Trans. on Consumer Electronics, pp. 285-297, May 1988.

[2] M. Yamashina et al.,"A Microprogramable Real-Time Video Signal Processor (VSP) for Motion Compensation", ITTT Trans. on Solid-State Circuits. 23:907-915, Aug. 1988.

1.2 Objetivos

É muito alta a necessidade de criação de um sistema digital de tratamento de sinais de vídeo em laboratórios de PDI, particularmente para o laboratório de PDI/DECOM. A disponibilidade de estações de trabalho - WorkStations -, dada a sua eficiência, satisfazem quase todos os requisitos anteriormente comentados; e possibilitam a formação de um sistema compacto sem a necessidade de implementação de hardware dedicado para os devidos fins. Restringimo-nos assim ao projeto e implementação de interfaces que possibilitam a digitalização de sinais de vídeo [3] e como tema deste trabalho a implementação da interface Homem-Máquina, Ambiente de Trabalho, responsável pela operacionalidade do sistema.

Este trabalho tem como objetivo a implementação de um Ambiente de Trabalho com o intuito de se criar um sistema de Digitalização e Processamento de Imagens - SDVI.

É utilizado como infra-estrutura estações de trabalho -WorkStations- dado os benifícios que a máquina proporciona. Usamos para a implementação características do ambiente de janelas OpenWindows seguindo portanto a padronização "OpenLook". [17], [18][17].

Este ambiente dado o seu objetivo, une várias ferramentas já existentes de PDI, e permite o uso através de suas opções, de trabalhos de processamento elaborados pelo grupo de vídeo do DECOM/FEE/UNICAMP bem como de pacotes de software públicos como o Khoros (The Khoros Group, University of New Mexico - E.U.A.) e o XV (Version 2.21, John Bradley, University of Pennsylvania, 1992).

Em particular, dada a necessidade da existência de funções de conversão de freqüências, outro objetivo deste trabalho foi a implementação das funções correspondentes para conversão, usando-se para isso o teorema de amostragem [3].

Este trabalho se resume em cinco capítulos

O capítulo 2 tem como objetivo, apresentar o Devguide, ferramenta usada para a elaboração do ambiente **SDVI**. Inicialmente apresentam-se suas origens e a bibliografia necessária. São apresentados vários elementos componentes da ferramenta, suas especificidades, importâncias e aplicações. Por último, apresenta-se o gerador de códigos e o compilador de fontes. O capítulo tende a elucidar uma ferramenta de grande importância, mostrando as facilidades de uso da mesma.

O capítulo 3 dá inicialmente ênfase ao Sistema Digital de Visualização e Processamento de Imagens - SDVI. Descreve sua importância e a necessidade de sua criação para uso em laboratórios de PDI e, em particular para o grupo de PDI/DECOM. Expõe sucintamente uma proposta de implementação em Hardware do sistema em função dos dois sinais, PAL-M e NTSC a serem digitalizados. Em seguida, falamos sobre o ambiente de trabalho do SDVI desenvolvido, principal tema deste trabalho, sua composição, janelas e atribuições, funções já existentes anexadas, pacotes de software públicos com o Khoros e o XV, descrevemos a funcionalidade e por último a maleabilidade do sistema.

O capítulo 4 expõe a implementação de funções de conversão de freqüência de sinais de imagens em componentes RGB. Usando-se a teoria de amostragem foram implementados programas em linguagem C, feitas simulações através de sinais de teste apropriados, foram determinados certos parâmetros e critérios para a elaboração e otimização das funções de conversão dos

[3] Outro tema de tese do grupo de PDI/DECOM

sinais. Os sinais de teste são avaliados pela relação sinal/ruído e, apresentamos posteriormente imagens processadas juntamente com imagens originais para uma avaliação subjetiva.

O capítulo 5 é uma conclusão do trabalho elaborado.

Apêndices dos programas são anexados no final do trabalho.

Finalmente, desejamos fazer uma ressalva quanto aos termos técnicos usados. Uma grande parte desses termos estão em inglês (sem aspas) e decidimos manter a nomenclatura original a fim de evitar confusões (já que esses termos aparecem dessa forma nas estações usadas em nosso trabalho).

Capítulo 2

OpenWindows Developer's Guide1.1

2.1 Introdução

O OpenWindows Developer's Guide 1.1 [1] é uma ferramenta projetada para facilitar o desenvolvimento de Interfaces. Referenciado como Devguide opera dentro do ambiente OpenWindows, interface gráfica usada em Estações de Trabalho Sun. O Devguide permite a criação de interfaces seguindo a padronização OPEN LOOK User Interface^{TM} [17], [18] como implementada para o OpenWindows. A palavra "Guide" em "OpenWindows Developer's Guide" é uma abreviação para "Graphical User Interface Design Editor".

O Devguide é usado para fazer a montagem dos elementos pertencentes a uma interface, através do manuseio de representações visuais destes da janela principal, Fig.2.1, para a "*workspace*" da estação. Na interface em construção, podem-se associar janelas, áreas de controle, botões, menus e outros elementos pertencentes a OPEN LOOK UI e, inclusive, testar a operação dos elementos associados ainda em plena montagem, sem a necessidade de compilação. Associados os componentes, a interface é salva em um arquivo que poderá ser chamado e modificado quando necessário ou, usado para gerar o código fonte. O Gxv (Devguide's companion program) parte do pacote Devguide faz o trabalho de codificação, lendo o arquivo criado, gerando chamadas para XView e fontes em C necessárias para criar a interface em OpenWindows. O Gxv gera ainda três arquivos: o Makefile, o Header, e o Stubs que ajudam o programador a unir os códigos da interface com os códigos aplicativos principais.

O Devguide roda em qualquer estação Sun em que se esteja usando o OpenWindows e o SunOS. O OpenWindows proporciona a interface gráfica, sistema de janelas e, a ferramenta necessária para rodar o Devguide e compilar as interfaces Devguide. O SunOS proporciona o compilador C e o ligador necessários para compilar e ligar as interfaces Devguide. Para o uso do Devguide é aconselhável a familiaridade com o UNIX, SunOS [12], OpenWindows[11], OPEN LOOK UI [17] [18], XView [2] e programação em linguagem C[13].

2.2 Inicializando o Devguide

Para inicializar o Devguide, deve-se primeiro rodar o OpenWindows. O Devguide pode então ser inicializado pelo gerenciador de arquivos ou pela janela de ferramentas Shell. Para a inicialização pelo gerenciador de arquivos é necessário que se esteja dentro do diretório "guide"; o Devguide provavelmente estará no subdiretório "bin". Pressione duas vezes sobre o ícone do Devguide com o botão SELECT do "mouse". A janela principal do Devguide como mostra a Fig.2.1 aparecerá na área de trabalho da estação (workspace). Pela opção Shell, escreva "*guide*" depois do "prompt" e pressione ENTER. Para finalizar o Devguide quando em operação, escolha a opção "quit" do menu da janela principal do Devguide. Para abrir esse menu, posicione o ponteiro do "mouse" no topo da janela Devguide, região identificada com o nome "OpenWindows Developer's Guide -1.1", e pressione o botão MENU do mouse.

2.3 Composição da janela Devguide

A janela Devguide contém controles e informações necessárias para operar o Devguide e os blocos ("glyphs") dos elementos necessários a construção das interfaces. Os controles estão localizados no topo, os blocos no meio e as informações acerca dos elementos na base da janela respectivamente.

I- Os controles da janela Devguide são:

File, View, Edit, Properties e Build/Test que contêm menus com as opções conforme Tab.2.1.

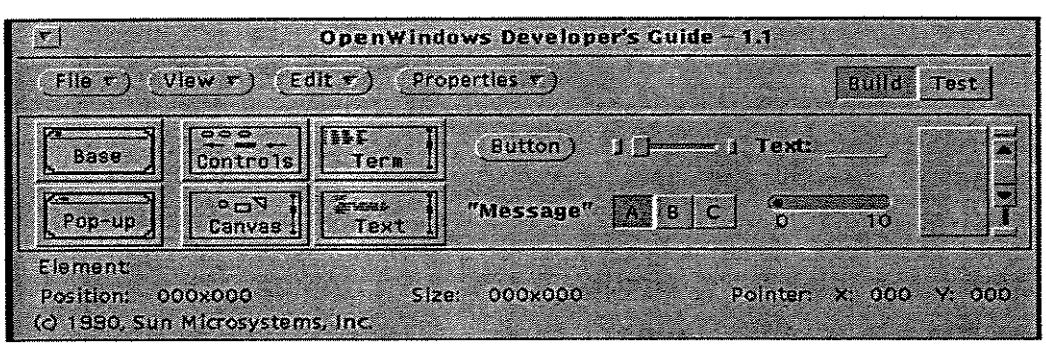

Figura 2.1: Janela principal do Devguide

• **Properties** Contém um menu com ítens que abrem janelas para edição de propriedades de elementos individuais dentro da interface do usuário. Veja Fig. 2.2.

Devguide - Controles				
File	View	Edit	Properties	Build/Test
Load Save Save As Close	Dismissed Window	Cut Copy Paste Delete Undo		

Tabela 2.1: Controles da janela Devguide

• **Build/Test** Opções para colocar o Devguide no modo montagem ou teste. Com a opção *"default" Built*, os elementos são associados de forma a se criar a Interface do Usuário (UI). Uma vez criada, a opção *Test* permite simular a interface manipulando sua estrutura funcional, sem que haja qualquer tipo de interação com os programas fontes a adicionar para o ambiente.

II- Informação acerca dos elementos:

A informação acerca dos elementos é mostrada na base da janela e tem os campos: **Element, Position, Size e Pointer.**

Para mais detalhes consulte "Apêndice Devguide".

III- Os elementos da Interface do usuário:

No centro da janela Devguide, encontram-se representados em miniatura os blocos dos elementos que podem ser adicionados à interface do usuário.

Selecionando os elementos da interface do usuário

Esses elementos, como anteriormente comentado, podem estar no modo de Edição ou no modo de Teste. Quando no modo de teste, estes funcionam como se estivessem em uma interface de um programa já concluído, permitindo a simulação do ambiente em criação, sem que haja interação com os programas fontes a incluir. No modo de edição, os elementos que compõem a interface são escolhidos um a um e montados, segundo as conveniências do usuário. Para posicionar um elemento "UIElement" na área de trabalho, pressiona-se o botão SELECT do mouse sobre o mesmo e arrasta-se o elemento para a posição desejada da *Workspace*. Alguns elementos, como janelas e paineis, precisam ser dimensionados assim que selecionados. Para tal, pressiona-se o botão SELECT do mouse em um dos cantos puxadores, levando-o para a posição que proporciona a dimensão desejada. Para o posicionamento do elemento na área de trabalho, simplesmente, feita a seleção do mesmo, arrasta-se para o lugar desejado a menos que o elemento seja uma área de controle ou um painel.

A seleção de um elemento é uma ação importante. O elemento deve ser selecionado para que se faça o processo de edição. A seleção é feita pressionando-se o botão SELECT sobre o elemento na área de trabalho, ou abrindo-se a janela de propriedades para esse tipo de

elemento e pressionando-se SELECT no nome do elemento na lista "scrolling list" do topo da janela, Veja Fig.2.3 para o elemento *base window*. Uma vez selecionado o elemento, uma caixa de seleção aparece por volta deste na área de trabalho e uma outra aparece em seu nome na lista do topo da janela de propriedades. Usando-se o botão ADJUST, vários elementos podem ser selecionados ao mesmo tempo.

Abrindo uma janela de propriedades para o elemento

Para editar as propriedades de um elemento o usuário deve selecionar primeiro o elemento (UIE) e depois abrir a sua janela de propriedades. Dentro da janela de propriedades existem vários campos de edição, onde o usuário define as propriedades do elemento.

A janela de propriedades para um elemento pode ser aberta de três maneiras:

- Selecionando o elemento e depois escolhendo "Properties" de seu menu;
- Escolhendo o tipo de elemento através de "Properties"da janela principal do Devguide e selecionar o nome do elemento na lista existente no topo da janela de propriedades que se abre;
- Pressionando-se duas vezes o elemento.

Elementos da Interface: User Interface Elements - UIE

Existem vários elementos, como podemos observar no centro da Fig.2.1, temos: **Base Windows, Pop-up Windows, Control Areas, Canvas Panes, Term Panes, Buttons, Messages, Settings, Text Field, Sliders, Gauges e Scrolling Lists.**

Base Windows

No OpenWindows, essa janela, define uma seção da área de trabalho, onde um aplicativo fornece informações, executa funções e oferece controles para o usuário. As interfaces devem ser inicializadas através de uma *Base Window.*

Pop-up Windows

No OpenWidows uma *Pop-up Window*, como a *Base Window*, define uma seção da área de trabalho onde um aplicativo fornece informações, realiza funções e, oferece controles para o usuário. A janela *Pop-up* é uma auxiliar para a *Base Window* e nunca aparece como principal em um programa.

Control Areas

Uma área de controle no OpenWidows, *Control Area*, é a região de uma janela onde o programa exibe controles como os: "*Buttons, Settings Sliders ...*". A área de controle deve ser colocada dentro de uma janela e sobre ela os controles.

Canvas Panes

No OpenWidows, *Canvas Pane* é a região de uma janela onde o programa apresenta gráficos. O usuário pode exibir nessa região seus próprios gráficos ou editar os já existentes.

Term Panes

No OpenWidows, *Term Panes* é a região de uma janela onde o programa apresenta o *UNIX Shell* para o usuário. Apresenta o *prompt* onde o usuário escreve comandos UNIX e trabalha diretamente com o sistema operacional *SunOS.*

Text Panes

No OpenWindows a *Text Pane* é uma região da janela onde o programa exibe texto e onde, por conseguinte, o usuário pode editar novos textos. Para que o usuário possa correr ao longo do texto contido na *Text Pane*, existe um cursor posicionado verticalmente no lado direito da tela.

Buttons

No OpenWindows, o *Button* é um botão de controle dentro da área de controle, que inicializa uma ação ou apresenta um menu se pressionado. Quando se é amarrado um menu para o botão ele é denominado *menu button.*

Messages

No OpenWindows, *Messages* é um controle que exibe mensagens para o usuário em texto no modo de leitura ou como uma imagem. Pode se usar *message* para adicionar textos ou imagens na área de controle.

Settings

O *Setting* é um controle que oferece diferenciadas alternativas de escolha de funções.

Text Field

Text Field é um controle no OpenWindows, que solicita do usuário um conjunto de palavras na forma de texto ou apresenta uma série de palavras para que o usuário edite em determinado campo. São tipicamente usados para pedir valores numéricos ou arquivos acompanhados de seus respectivos diretórios. Somente pode ser colocado dentro de áreas de controle.

Sliders

No OpenWindows, *Sliders* é um controle dentro da área de controle que posiciona um cursor em determinado valor dentro de uma faixa de valores. Para a escolha de um valor, o usuário corre com o cursor, usando para isso o mouse, ao longo da faixa de dados existente.

Gauges

Ao contrário de *Sliders*, o *Gauge* fornece para o usuário, determinado valor dentro de uma faixa de valores possíveis. É simplesmente um processo de leitura. Esse elemento é usado dentro de uma área de controle.

Scrolling Lists

Apresenta para o usuário uma lista em forma de texto na posição vertical, corrida por um cursor com ítens que o usuário pode escolher usando o ponteiro do mouse. É tipicamente

usada para acomodar confortavelmente um conjunto amplo de opções em forma de texto em um espaço limitado.

Para mais detalhes consulte o Apêndice Devguide. Nele encontramos a utilidade e as propriedades de cada elemento de interface.

Menus

Além dos elementos de controle e para controle que existem na região central da janela principal do Devguide, Fig.2.1, existe a opção menu criada através da janela de propriedades. O menu é uma janela do tipo *Pop-up* com várias opções de controle. Existem dois tipos de menus:

- *Pop-up menu:* Quando chamado, aparece em algum lugar da área de trabalho, *WorkSpace*, não amarrado a algum controle e pode ser atribuído a vários elementos de interface.
- *Button menu:* É amarrado a um botão. Esse tipo de menu usa uma marca no botão de procedência, que permite que o usuário perceba sua origem. Essa janela é aberta quando o usuário pressiona o ponteiro do mouse sobre o botão de origem.

As opções de escolha atribuídas a um menu são chamadas de ítens. Podem ser em forma de texto ou imagem. Os ítens podem: Inicializar uma operação ou abrir um sub-menu. Aos menus no OpenWindows, se podem atribuir várias camadas de sub-menus conforme necessidade do usuário - Sub-menus podem ter sub-menus que podem ter sub-menus indefenidamente. Qualquer item que abre um menu, possui uma marca característica de abertura desse tipo de janela.

2.4 O Gerador de Códigos Fonte

Uma vez criada, a interface é salva como arquivo **GIL** . Usa-se em seguida o Gxv, gerador auxiliar de códigos do programa, para criar os códigos fonte da interface do usuário. A utilização do gerador de códigos, escrita no prompt do sistema operacional *SunOs* , é seguida do nome do arquivo GIL, cuja extensão é **".G"** *(: gxv nome-do-arquivo GIL; sem escrever a extenção).*

Realizada a operação, o Gxv gera códigos fonte em liguagem C para cada elemento de interface de usuário criada. Armazena o código fonte e outras informações em quatro diferentes arquivos localizados no diretório corrente, são eles:

- **Main Source File**, arquivo fonte principal. Contém código fonte em C que gera os elementos da interface. Tem extenção "-ui.c".
- **Header File**, arquivo que contém diretrizes para os elementos do arquivo fonte principal. Tem extensão "-ui.h".
- **Notify Handler File**, arquivo de notificações. Contém inicialização de funções específicas as quais se inserem o código fonte em liguagem C, apropriado. Tem extensão "-stubs.c".

- **Help Text File**, arquivo texto para ajuda. Contém o texto gerado para mensagens de auxílio dos diferentes elementos de interface. Tem extensão "-info".

  O gxv, cria também o arquivo Makefile. Este arquivo é usado para fazer a ligação entre os códigos durante a compilação.

2.4.1 Compilação dos Códigos Fonte da Interface do Usuário

Para a compilação dos códigos fonte gerados para a interface, usa-se o comando *make*, seguido do nome do código do objeto que se deseja criar. O comando make, compila e liga os códigos, seguindo instruções contidas no makefile. Produz um arquivo objeto com o nome especificado pelo usuário.

Mais pormenores sobre os códigos gerados, os parâmetros existentes no Makefile e as alterações necessárias nesse arquivo, a forma de como fazer atribuições de objetos e suas inter-dependências podem ser encontradas na referência [1].

2.5 Conclusões

Em síntese, o capítulo apresenta a ferramenta **Devguide**. Mostra suas funções, elementos e informações decorrentes do processamento, a facilidade de uso e importância como ferramenta de suporte para a elaboração de interfaces gráficas.

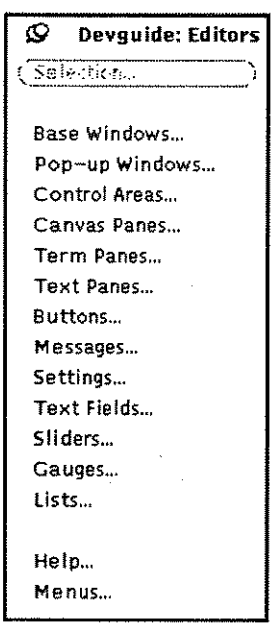

Figura 2.2: O menu de propriedades.

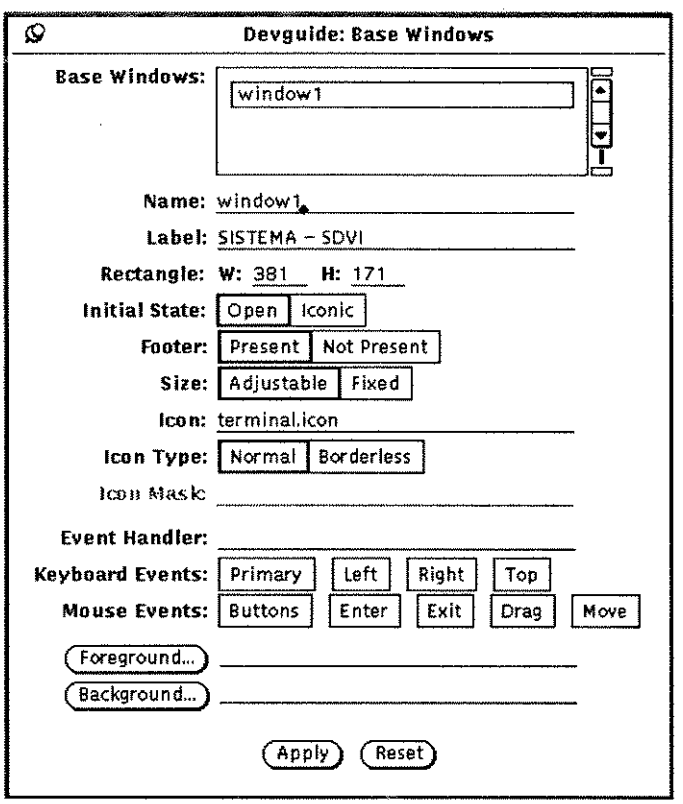

Figura 2.3: Edição de uma janela

Capítulo 3

Sistema Digital de Visualização de Imagens - SDVI

3.1 Introdução

O processamento de imagens exige fontes com uma grande quantidade de dados quando comparado com outros tipos de informações digitais, tais como de voz em telefonia. Essa quantidade tende a aumentar quando se trabalha com uma sequência de imagens paradas para compor partes de uma cena em movimento. Deve-se lembrar que o tratamento de sinais componentes de uma única imagem parada requer três arquivos, correspondentes a um sinal de luminância e a dois de diferença de cor. Uma sequência de dois quadros de televisão gera assim 6 arquivos fontes que devem ser manipulados e o resultado processado ocupa mais 3 arquivos de dados resultando num total de 9 arquivos de dados que devem ser armazenados. Os dados correspondem a amostras digitalizadas em 8 bits sendo que numa linha horizontal, pode-se ter de 606 (TV comercial) a 1920 (HDTV) amostras. Assim, o arquivo contém mais de 2 Mbits de dados de informação. Em geral, necessita-se de estatísticas correspondentes a pelo menos 4 imagens paradas (ex.: SMPTE - Society of Motion Picture and Television Engineer - SMPTE 01/Praia; SMPTE 02/Sala; SMPTE 04/Zelda; SMPTE 15/Cozinha), o que resulta em uma quantidade muito grande de dados a serem manipulados. Numa sequência de imagens em movimento, trabalha-se com cerca de 32 campos sucessivos que podem ser lidos 2 a 2 usando-se neste caso o auxílio de memória de massa.

Os parâmetros objetivos de medida (relação sinal/ruído, taxa de bits de transmissão, correlação, entropia, médias e variâncias) são importantes para a avaliação do desempenho de esquemas de processamento de imagens. A avaliação objetiva é constituída de cálculos que se traduzem na obtenção de parâmetros que refletem o comportamento do sinal nas várias etapas do processamento, e de outra parte confirmada através de testes e medidas experimentais. A avaliação subjetiva tem por finalidade medir a qualidade visual da imagem processada. É imprescindível, devido ao fato de que a imagem bidimensional pode ter defeitos perfeitamente visíveis e localizados numa região espacial restrita, embora a relação sinal-ruído seja tão boa quanto numa outra imagem em que a degradação se encontra numa região imperceptível de muitos detalhes ou distribuída por toda a imagem. Para a avaliação subjetiva, por outro

lado, faz-se uma comparação mostrando-se em monitores as imagens processada e original e atribuíndo-se notas que as qualifiquem em termos do grau de degradação da qualidade subjetiva da imagem processada. Para tanto, deve-se usar um elenco de observadores que atribuirão, por exemplo, notas de 1 a 5, conforme sugerido pelo CCIR. A opinião média traduz o desempenho do método de codificação em questão.

Um outro fator importante na avaliação subjetiva é a fidelidade quanto a reprodução das cores. Neste aspecto tem-se a limitação física relativa aos fósforos primários dos cinescópios e mesmo quanto ao número de cores simultâneas que podem ser reproduzidas em monitores de terminais de vídeo; no entanto, a avalição e a obtenção de resultados são agilizados quando se é possível uma comparação visual entre imagens processadas concomitamente com a obtenção dos valores numéricos. Assim, a validação do esquema deve necessáriamente ser submetida a ambos os critérios de avaliação: **objetiva e subjetiva.**

O projeto do Sistema Digital de Visualização de Imagens (SDVI) abrange pesquisas em "software" e "hardware" visando obter uma infra-estrutura computacional para aquisição e visualização de imagens digitalizadas, bem como subsídios para a realização de simulações de sistemas de codificação de imagens. As atividades que compõem o trabalho envolvem o desenvolvimento de partes do sistema, fornecendo subsídios para a definição técnica do sistema SDVI bem como das várias partes integrantes, tanto a nível de hardware como de software. As atividades de simulação para quais o sistema SDVI está sendo preparado, envolvem a codificação composta e de componentes de sinais de vídeo (ex.: TV convencional, videofone, teleconferência) e de imagens estáticas (ex.: "frozen pictures").

Do ponto de vista técnico, um dos nossos objetivos é a aplicação de técnicas e métodos de extração de redundâncias para a codificação de sinais-fonte de imagens. A motivação de tais processamentos é a tendência atualmente crescente de digitalização dos canais de transmissão de informação. Como já é sabido, no mundo moderno, a comunicação digital vem ocupando um espaço cada vez maior, pelas vantagens de tratamento e de armazenamento.

No extenso campo das imagens, por exemplo, no caso da TV comercial, a digitalização de sinais já é um fato dentro do estúdio, principalmente para mesas de corte, geração de efeitos e de gravações. Assim, mesmo a nível de tratamento, a forma digital tem permitido manipulações impossíveis na forma analógica, e coloca à disposição recursos visuais cada vez mais explorados para edição e radio-difusão. Para transmissão, a codificação digital aplicada à telefonia e comunicação de dados tem aumentado a capacidade dos canais em termos de velocidade, qualidade e diversidade. Rotas de alta hierarquia em 34 Mbits/s via cabo fibra ótica, rádio terrestre/satélite, tornam-se realidades de nosso dia a dia, praticamente exigindo a codificação digital do sinal de TV comercial, ainda que pelo simples motivo da inviabilidade de manutenção dos antigos equipamentos analógicos dedicados a este fim. Além disso, a rede de serviços integrados, juntamente com os serviços de multimeios deverão ser oferecidos pelas operadoras devido ao crescente interesse pelos mais diversos tipos de aplicações e necessidades. No Brasil, já se cogita de canais de 140 Mbits/s que potencialmente podem permitir o tráfego do sinal de alta definição digital de televisão, como testado no exterior. As taxas mais baixas são cogitadas para intercomunicação através de sinais de videofone e teleconferência (px64 Kbits/s,p=1 a 32). Taxas de modens até 19,2 Kbit/s são viáveis para transmissão de imagens estáticas, bem como de fac-símiles e similares.

O Sistema Digital de Visualização de Imagens vem para satisfazer a necessidade existente em centros de pesquisa no que toca à aquisição e processamento de imagens.

Os módulos básicos do sistema têm como objetivo:

- Digitalizar sinais de vídeo nos formatos compostos (Y+U+V) e componentes (RGB) para os sistemas em cores PAL/NTSC, padrão M. Os sinais analógicos são fornecidos por câmeras de TV comerciais. Naturalmente também podem ser usados equipamentos de VT.

- Reproduzir imagens em formatos padrões usando-se monitores tipo multisync, capazes de fornecer imagens com qualidade para permitir avaliações subjetivas.

Esse sistema de acordo com seu poder de processamento, e os devidos fins a que se destina permitirá ainda:

a) Armazenar até 32 campos de quadros sucessivos de imagens de TV convencional correspondentes a cenas em movimento;

b) Realizar simulações de esquemas de codificação de sinais de vídeo e imagem para avaliação objetiva do desempenho quanto ao compromisso entre o tempo de transmissão e a qualidade de imagem.

A composição do sistema pode ser descrita em dois módulos príncipais: **Hardware e Software.**

3.1.1 Hardware - Composição em blocos do sistema

Este módulo visa ilustrar a composição "Hardware" do sistema.
O sistema é composto basicamente pelos seguintes blocos:

1. Estágio de Digitalização - Os sinais analógicos de entrada (Tab.3.1) estão no formato PAL-M/NTSC ou de componentes RGB. A interface PAL-M (Fig. 3.3) é implementada à parte;

2. Estágio de Visualização - Os dados disponíveis no barramento ou em memória devem ser coletados para reprodução no monitor;

3. Memória - O equipamento provê memória de quadros da sequência em movimento. O "buffer" necessita de controle;

4. CPU - Um microcomputador (estação de trabalho) permite o comando das funções (digitalização, processamento, visualização) bem como a escolha das opções (YUV, RGB);

5. Lógica de Controle - Circuitos de sequênciamento.

As Figs. 3.1 e 3.2 mostram o digrama de blocos do sistema proposto.

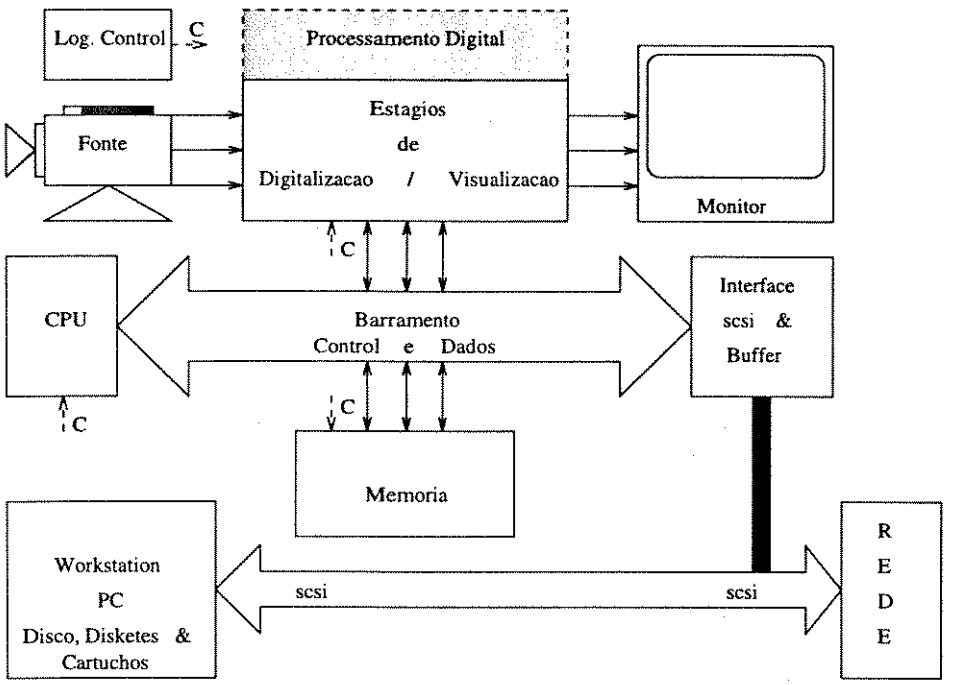

Figura 3.1: Modulos básicos do sistema.

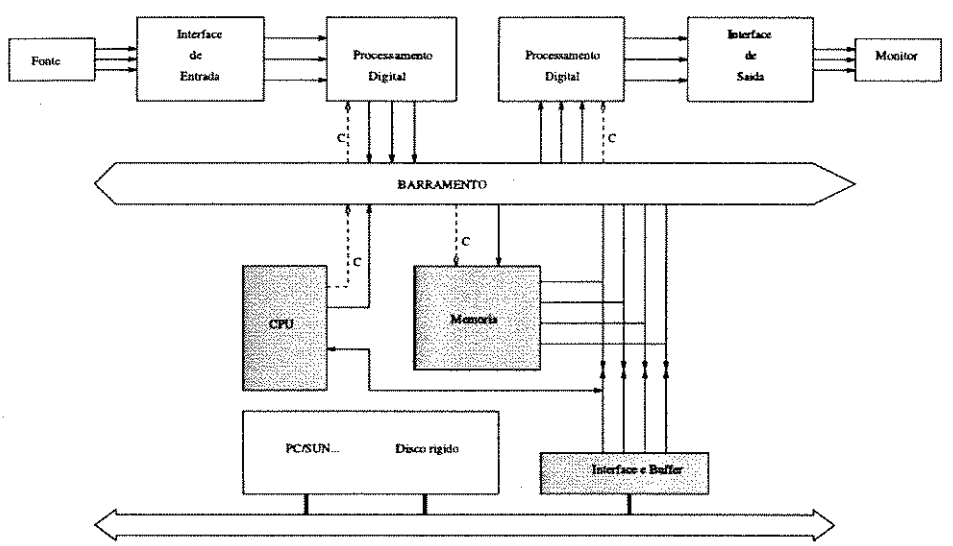

Figura 3.2: Estrutura Básica do Sistema.

SISTEMA PAL-M		SISTEMA NTSC	
PARÂMETRO	VALOR	PARÂMETRO	VALOR
Sinal de Luminância	$E' = 0,299E'_R + 0.587E'_G + 0.114E'_B$ onde "'" indica correção gama (γ) no caso igual a 2,8	Sinal de Luminância	$E' = 0,299E'_R + 0.587E'_G + 0.114E'_B$ onde "'" indica correção gama (γ) no caso igual a 2,2
Sinais Diferença de Cor	$E'_U = 0,493(E'_B - E'_Y) = 0,493C_B$ $E'_V = 0,877(E'_R - E'_Y) = 0,877C_R$	Sinais Diferença de Cor	$E_I = -0,27(E'_B - E'_Y) + 0,74(E'_R - E'_Y)$ $= -0,27C_B + 0,74C_R$ $E'_Q = 0,41(E'_B - E'_Y) + 0,48(E'_R - E'_Y)$ $= 0,41C_B + 0,48C_R$
Equação do Sinal Composto	$E_M = E'_Y + E'_V \sin(\omega_{SC}t)+$ $m(t).E'_V \cos(\omega_{SC}t)$ onde : $\omega_{SC} = 2\pi f_{SC}$ $f_{SC} = (3.575.611,49 + 10,00)Hz$ $m(t) = \pm 1$ com período $2T_H$	Equação do Sinal Composto	$E_M = E'_Y + E'_Q \sin(\omega_{SC}t + 33)+$ $+E'_I \cos(\omega_{SC}t + 33)$ onde : $\omega_{SC} = 2\pi f_{SC}$ $f_{SC} = (3.579.549 + 10)Hz$
Fase da Sub-portadora	$s(t) = -K.0,707[-\sin(\omega_{SC}t)+$ $+m(t)\cos(\omega_{SC}t)]$ onde: K = 150mV	Fase da Sub-portadora	$s(t) = -K \sin(\omega_{SC}t$ onde K = 143mV
Intervalo Apagamento de Salva	11 linhas sendo: 260 a 270 522 a 7, 259 a 269, 523 a 8	Intervalo Apagamento de Salva	compreendem 19 $T'_H s$ a partir do 0v
Freqüência da Subportadora de cor	$(909/4)f_H$ onde: f_H = freqüência de linha	Freqüência da Subportadora de cor	$(455/2)f_H$ f_H = freqüência de linha

Tabela 3.1: Características dos sinais de vídeo PAL-M e NTSC

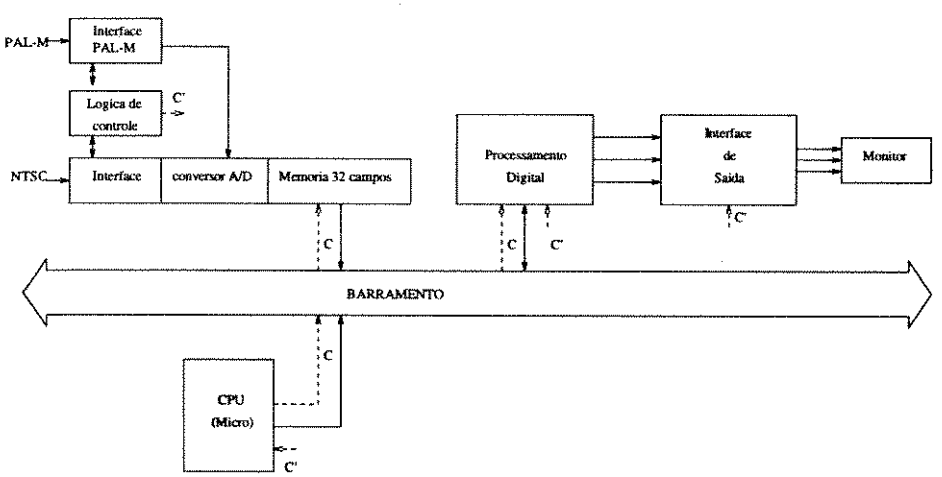

Figura 3.3: Composição do sistema - interfaces

3.1.2 Software - Software básico

Uma estrutura em software forma o ambiente de interação do usuário com o sistema. Essa estrutura dá ainda suporte às principais funções internas de manipulação do SDVI.

As funções formam três grupos principais: Aquisição, Processamento e Visualização.

Função de Aquisição: A estrutura deve ser tal que permita a escolha entre:

- As diversas opções de formatos de entrada analógica;
- Definição do padrão;
- Freqüência de amostragem (8/3fsc, 13.5Mhz e 4fsc);
- Sistema em cor (PAL, NTSC);
- Número de bits por amostra;
- Com ou sem APH;
- Com ou sem APV;
- Número de linhas de varredura, até 263 linhas/campo;
- Com ou sem salva de portadora de cor;
- Tamanho da matriz (512x512 default; $n_l \mathrm{X} n_c; n_l = 1a525, n_c = 1a909$);
- Número de campos por quadro;

Estas, são as opções básicas de manipulação do digitalizador, que como proposta de projeto para interfaces de entrada, inclui em sua estrutura as interfaces NTSC e PAL padrão M.

Funções de Processamento: Incluem-se diversas funções destinadas para:

- Realizar operações de filtragem diretamente nos arquivos de imagem;
- Armazenar em memória determinada imagem presente no barramento;
- Realizar operações de manipulação geométrica, extração de características, conversão de formatos de arquivos, álgebra de matrizes, manipulação de histogramas, operações estatísticas, lineares, transformadas, geração de sinais.

As filtragens visam obter deslocamentos, composições para mudança da freqüência de amostragem, eliminação de ruído e geração de amostras correlatas ou não.

Função de Visualização:O sistema deve manipular através dessa opção o processo de visualização de imagens. Estas podem ter origem em:

- Arquivos já existentes;
- Da interface de entrada, armazenada em 1 ou vários (até 32) campos do buffer de digitalização, obtendo-se com vários campos em sequência imagens contínuas de curta duração;

- Um arquivo de estação SUN, PC ou vindo através da interface do sistema de outro equipamento de acesso remoto.

As opções incluem os formatos de saída, freqüência de amostragem, número de campos/quadros sucessivos, número de bits. Outras opções devem permitir a escolha de sinais Y,U,V; R,G,B; Y+U+V ou RGB ou, U+V entre outras opções.

Em suma, estas funções dão origem à diversas opções de tratamento do sinal.

3.2 O Ambiente de Trabalho do SDVI

3.2.1 O Ambiente SDVI

O Ambiente de Trabalho SDVI, surge como necessidade de uma ferramenta capaz de proporcionar recursos para o processamento de sinais de vídeo. O sinal de vídeo é digitalizado à uma freqüência de 4fsc, analisado por múltiplas formas através de opções que o ambiente possui e, por fim, visualizado ou transmitido através de rede para um ponto remoto. Procurou-se trabalhar para a criação de um ambiente interativo de fácil manuseio, com explicações *("helps")* que possibilitam maior acesso aos recursos existentes. Além disso, com a idéia de abrangência, foram estudadas e incluídas opções que permitem adicionar futuramente funções necessárias ao bom desempenho do sistema. Dessa forma, atribuiu-se ao sistema características de adaptação ao estado de arte vigente, ampliável conforme necessidade do grupo. O ambiente segue as características da padronização OPEN LOOK User Interface^{TM} [17] e possui as facilidades do ambiente gráfico de janelas OpenWindows [11]. É formado por uma série de janelas interligadas de forma a satisfazer a operação desejada.

A janela principal do SDVI (Fig. 3.4) é composta pelas opções:

A) Aquisição...

B) Processamento...

C) Visualização...

D)Informações Gerais...

Inicialmente a janela aparece no seu estado de "aberta" conforme a Fig.3.4. Quando necessário, pode ser fechada para o seu Ícone correspondente Fig. 3.5. Essa transformação é feita através da máscara existente na extremidade superior esquerda da janela ou por intermédio da opção do *"Menu"* conforme características do OpenWindows. Inclui-se para este último caso a operação inversa -Ícone para janela.

Resumidamente, as opções permitem:

Aquisição:

Digitaliza e Armazena o sinal de vídeo à uma freqüência de 4fsc (existe opção para outras freqüências);

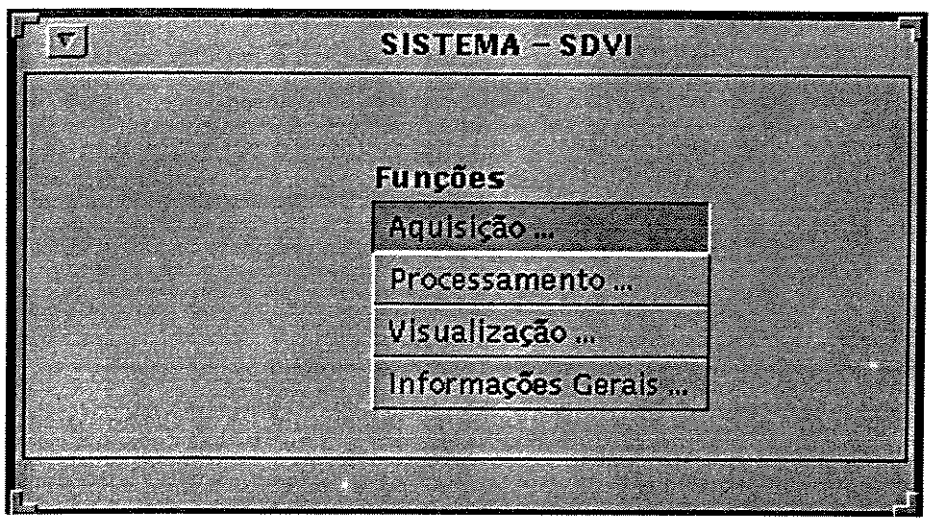

Figura 3.4: Janela principal do SDVI

Figura 3.5: Icon - SDVI

Processamento:
Processa o sinal de vídeo digitalizado através de múltiplas opções do sistema. Inclui-se neste caso a utilização de ferramentas de domínio público como o **Khoros** [16];

Visualização:
Usa ferramentas como o **Xview** e o **Khoros** para visualizar imagens em seus formatos correspondentes;

Informações Gerais:
Pretende-se formar com esta opção um banco de dados concernente ao material existente no Grupo de Imagens e pesquisas afins.

3.2.2 Aquisição

O processo de aquisição do sinal de vídeo digitalizado é feito através de interfaces A/D, segundo características da Fig.3.3. Apesar das várias opções existentes no Ambiente, inicialmente se dá preferência para a digitalização na freqüência de 4fsc (fsc: freqüência de subportadora de cor) devido as características de alinhamento regular do mosaico, numerosidade de amostras (permite maior informação de detalhes) e a facilidade de conversão para outras freqüências.

A janela para esse processo (Fig. 3.6), inclui as mais diversas opções com o intuito de se tornar abrangente a técnicas de digitalização futuras.

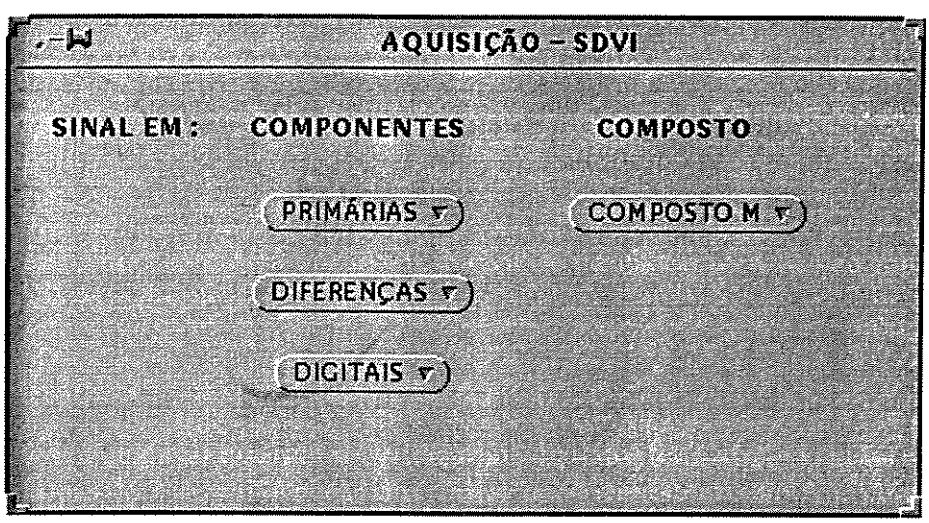

Figura 3.6: Janela do SDVI para AQUISIÇÃO

Inicialmente, o sinal pode estar em Componentes ou em seu estado Composto. Para isso dividiu-se a janela de Aquisição em duas partes: **Sinal em Componentes e Sinal Composto.**

Perceba que, como a janela é do tipo "Pop-up", para que permaneça na área de trabalho feita a operação desejada, é necessário que se faça uso (introduzir) do alfinete em sua extremidade superior esquerda.

Sinal em Componentes:

O Sinal em Componentes pode ser constituído pelas suas: Primárias RGB (PAL-M / NTSC / HDTV), Sinais Diferença YUV (PAL-M), YIQ (NTSC) ou pelos Sinais Digitais YCrCb (PAL-M / NTSC) e YPbPr (HDTV). Para isso temos as opções:

1. **Primárias**;
2. **Diferenças**;
3. **Digitais**.

Primárias:

O sinal digitalizado em suas componentes primárias RGB pode dar origem aos sinais PAL-M, NTSC ou HDTV. Para isso criou-se um sub-menu de duas opções, uma que trata de sinais RGB em freqüências de 4fsc, 13.5Mhz, 10Mhz e 8/3fsc e outra para sinais RGB com nível de HDTV e digitalização a uma taxa de 70Mhz (Fig. 3.7).

Diferenças:

O sinal digitalizado em suas Componentes Diferenças dá origem ao sinal YUV ou YIQ. Para essa alternativa criou-se um sub-menu de duas opções PAL-M (YUV) e NTSC (YIQ) que através de outro sub-menu conforme Fig. 3.8, o sinal pode ser digitalizado usando-se freqüências de 4fsc, 13.5Mhz, 10Mhz e 8/3fsc.

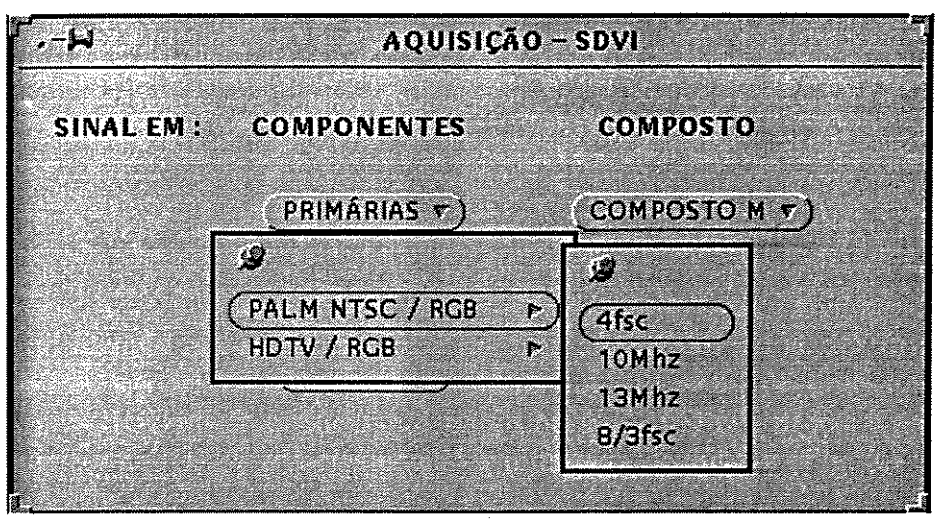

Figura 3.7: Aquisição em componentes primárias

Digitais:

Esta opção abre um sub-menu que permite a digitalização do sinal YCrCb (PAL-M / NTSC) ou YPbPr (HDTV) (Fig. 3.9). O sinal YCrCb pode ser digitalizado através de outro sub-menu que se abre e ao "Hardware" correspondente à freqüências de 4fsc, 13.5Mhz, 8/3fsc e 10Mhz. O sinal YPbPr é digitalizado a 70Mhz.

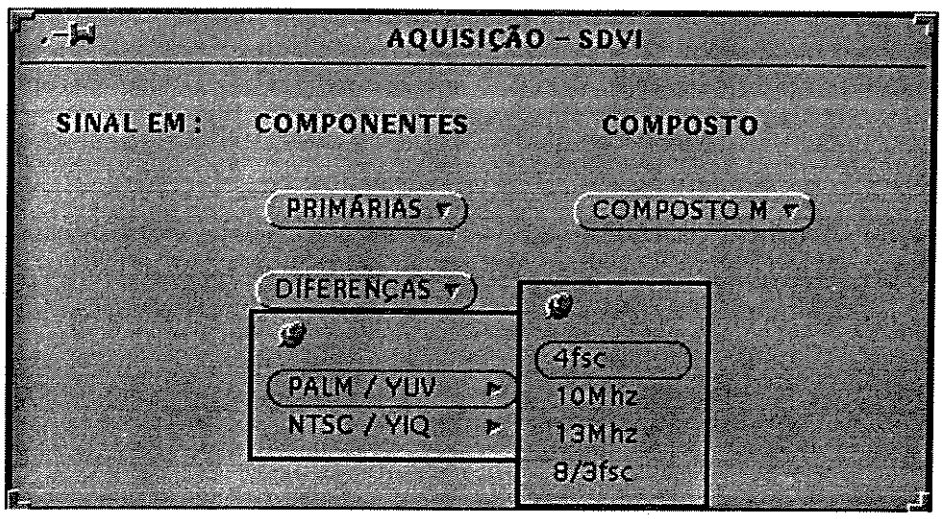

Figura 3.8: Aquisição em componentes diferenças

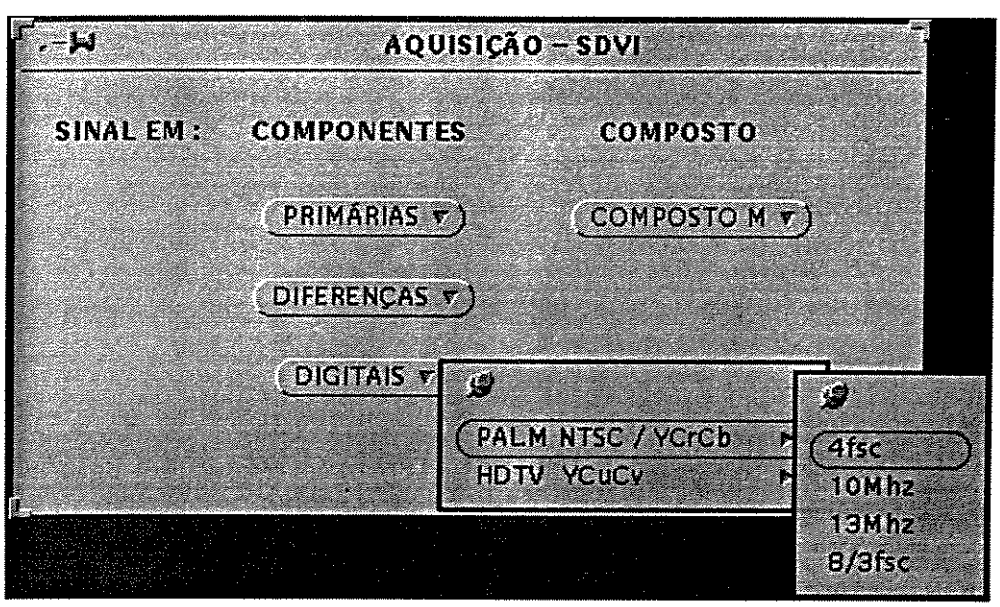

Figura 3.9: Aquisição em componentes digitais

Sinal Composto:

Nesta segunda forma de digitalização de sinais de vídeo o sistema abre um sub-menu com três opções:

1. **PAL-M**
2. **NTSC**
3. **HDTV**

que permitem a digitalização do sinal composto PAL-M, do sinal composto NTSC e sinal composto HDTV, respectivamente. As freqüências de digitalização aparecem através de um sub-menu do sistema correspondente escolhido e são: 4fsc, 13.5Mhz, 8/3fsc e 10Mhz para os sistemas PAL-M/NTSC e 70Mhz para HDTV (Fig. 3.10).

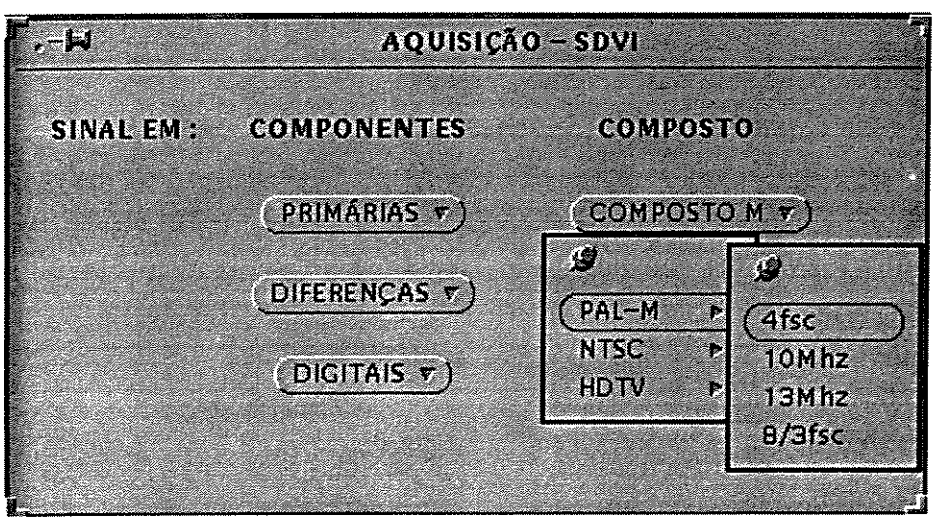

Figura 3.10: Aquisição sinal composto PAL-M

As freqüências usadas para a digitalização dos sinais de vídeo são as de características estudadas. As várias opções existem de forma que se anexem ao sistema novas tecnologias de digitalização. Através do botão do "mouse" o usuário escolhe a freqüência de operação desejada, abrindo para isso vários sub-menus e, por conseguinte a operação de digitalização via "Hardware". Em todos esses processos de digitalização se dá preferência a taxa de amostragem de 4fsc já que se adotou como freqüência de referência para o SDVI.

3.2.3 Processamento

Esta opção tem como objetivo o processamento do sinal de vídeo digitalizado. Depois de digitalizado, o sinal é processado conforme necessidade do usuário. Existem vários pacotes de domínio público que oferecem ferramentas de processamento de imagens. Utilizando-se essas facilidades subdividiu-se a opção de processamento (Fig. 3.12) em três alternativas:

1. **Uso do Khoros**
2. **Outros Aplicativos**
3. **Uso do SDVI**

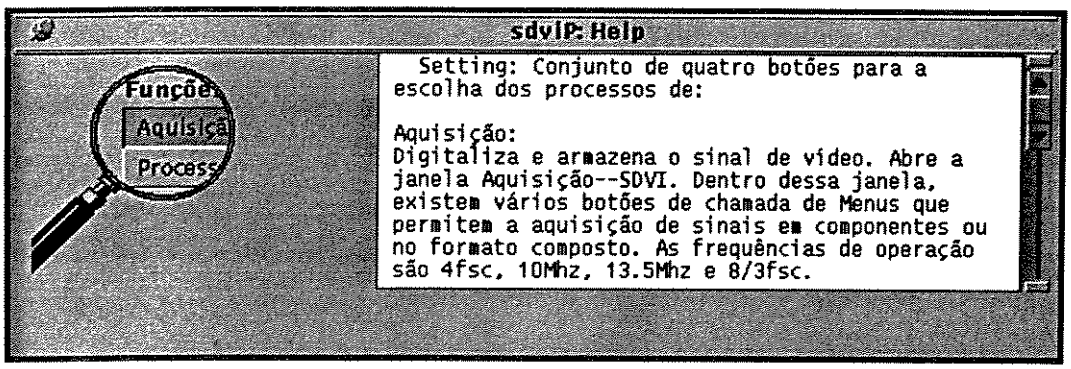

Figura 3.11: Exemplo de Janela do SDVI para Ajuda

Figura 3.12: Janela do SDVI para Processamento

Como tal passamos a descrever cada uma das opções.

Uso do Khoros:

Esta opção permite que se use o Khoros. O Khoros é uma ferramenta de domínio público - ("The Khoros Group, Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque") [16]. É um integrado de software e ambiente de desenvolvimento que se destina a processamento de informação e visualização de dados baseada no X11R4. Compõem o Khoros a linguagem de programação VISUAL, Geradores de Código para a linguagem VISUAL, o editor iterativo de interfaces para o usuário, um pacote para visualização de imagens, uma extensiva biblioteca para processamento de imagens, análises numéricas, rotinas de processamento de sinais e, pacotes para traçar gráficos 2D

e 3D.

Aplicações X Windows:

Animate - Ferramenta iterativa de exibição de sequências de imagens.
Cantata - Liguagem de Programação VISUAL extensível.
Concert - Sistema de distribuição de Interfaces X do usuário.
Editimage - Exibição iterativa de imagens e manipulação de programas.
Xprism2/Xprism3 - Traçadores de gráficos 2D e 3D.
Viewimage - Programa iterativo básico de interpretação e definição de superfícies.
Warpimage - Programa iterativo para registro e deformação de imagens.

Algorítmos para Processamento de Dados:

O Khoros contém mais de 260 programas nas seguintes categorias: aritmética, classificação, conversão de cores, conversão de formatos de arquivos, extração de características, filtros de freqüência, álgebra matricial, filtros espaciais, morfologia de filtros, manipulação geométrica, manipulação de histogramas, estatística, geração de sinais, operação linear, segmentação, estimativa espectral, subregiões e, transformadas. O Khoros dá suporte aos formatos de arquivos: TIFF, pbm, BIG, DEM, DLG, ELAS, MAT*AB, Sun raster, TGA e xbm.

Ferramentas de Interface do Usuário:

Preview - Ferramenta de exibição de interfaces gráficas do usuário.
Composer - Editor iterativo de interfaces gráficas do usuário.
Conductor - Ferramenta de geração de códigos para a interface gráfica do usuário.
Ghostwriter - Ferramenta de geração de códigos para a linha de comando da interface do usuário.
Source configuration e Management - Ferramenta para configuração e manutenção do sistema.

Para o processamento de imagens usando-se o Khoros, criou-se um menu de quatro opções que permite usar: O Cantata, Xprism2, Xprism3 e o Warpimage.

1.Cantata: Esta opção permite usar o Cantata. Um *glyph* é um pequeno ícone representando uma rotina de processamento, que pode ser selecionado da lista de subrotinas. O Khoros tem uma biblioteca de mais de 260 subrotinas nas categorias anteriormente mencionadas. Um programa *visual* consiste de um certo número de *glyphs* interligados. Os *glyphs* contêm pontos de conexões de entrada e saída, representados por botões localizados à esquerda e direita dos blocos, respectivamente. As conexões de um *glyph* a outro, são feitas pressionando-se com o "mouse" o botão de saída de um e o de entrada de outro (ou vice-versa). O programa é salvo em um arquivo temporário. Em geral usa-se esse arquivo como entrada para o próximo programa e assim por diante. O programa pode ser executado usando-se o botão 'RUN' localizado na janela ou, ser executado passo a passo pressionando-se para isso pequenas chaves 'ON/OFF' que aparecem na extremidade superior esquerda de cada *glyph*. Essas chaves representam um programa executável do Khoros. A qualquer momento, a *workspace* criada pelo Cantata pode ser salva como arquivo ascii. Posteriormente, através do Cantata, carrega-se a *workspace* salva, de forma que se dê continuidade ao programa *visual* [16].

2.Xprism2: É um pacote para traçar gráficos unidimensionais, para uso independente ou com outras partes do Khoros. Existem várias formas de entrar com dados: arquivos de imagens, arquivos de funções ou dados de pontos e, funções que têm entrada pelo teclado. O formato dos dados de entrada deve ser representado pelos pares XY ou somente Y. Para arquivos de dados *raw* deve ser selecionado o tipo de padrão para BYTE, se *Short, Integer* ou *Float*. Para os gráficos, se podem usar recursos de translação, rotação e mudança de escala. Pode-se ainda, trocar os eixos, as fontes, as cores, tipos de marcas, linhas, traçados e títulos. A informação do traçado pode ser salva implícitamente ou explicitamente em um arquivo de formato de imagem. A saída pode ser enviada para postscript, imagem, ou impressoras laser ou HPGL [16]. Ao se usar esta opção chama-se o Xprism2. Abre-se então uma janela com uma área onde será traçado o gráfico e a esquerda uma série de botões que permitem:

Plot	: Usado para criar e traçar gráficos
Options	: Usado para propriedades do gráfico
Annotation	: Anotações para o gráfico
Output	: Para enviar o gráfico à uma impressora ou salvar como arquivo .VIFF
Answer Files	: Usado para a leitura e escrita de arquivos *answer* Xprism2

Existem também oito botões de localizados na parte esquerda da janela, que permitem ao usuário obter informações acerca do gráfico e para realização de certas operações na *workspace*:

PLOT INFO	: Usado para obter estatísticas acerca do gráfico
REFRESH	: Usado para redesenhar a *workspace*
CLEAR	: Usado para limpar completamente a *workspace*
HIDE	: Usado para ocultar um ou mais traçados
DELETE	: Usado para apagar um ou mais traçados
RESET	: Usado para reset dos eixos do traçado
HELP	: Exibe a documentação do man Xprism2
QUIT	: Saída do Xprism2

3.Xprism3: É um pacote para traçar gráficos tridimensionais para uso independente ou com outras partes do Khoros. Os dados a serem traçados têm origem em: arquivos de imagens, arquivos de funções ou dados de pontos e, funções que têm entrada pelo teclado. O formato dos dados de entrada deve ser representado por XYZ ou apenas por Z. Para arquivos de dados *raw* deve ser selecionado o tipo de padrão para BYTE, se *Short, Integer* ou *Float*. Para os gráficos, podem-se usar recursos de translação, rotação e mudança de escala. Pode-se ainda, trocar os eixos, as fontes, as cores, tipos de marcas, linhas, traçados e títulos. A informação do traçado pode ser salva implícitamente ou explicitamente em um arquivo de formato de imagem. A saída pode ser enviada para postscript, imagem, ou impressoras laser ou HPGL [16]. Ao se usar esta opção chama-se o Xprism3. Abre-se então uma janela com uma área onde será traçado o gráfico e a esquerda uma série de botões que permitem:

Plot	: Usado para criar e traçar gráficos
Options	: Usado para propriedades do gráfico
Annotation	: Anotações para o gráfico
Output	: Para enviar o gráfico à uma impressora ou salvar como arquivo .V
Answer Files	: Usado para a leitura e escrita de arquivos *answer* Xprism2

Existe também oito botões de 'ação' localizados na parte esquerda da janela, que permit ao usuário obter informações acerca do gráfico e para realização de certas operações *workspace*:

PLOT INFO	: Usado para obter estatísticas acerca do gráfico
REFRESH	: Usado para redesenhar a *workspace*
CLEAR	: Usado para limpar completamente a *workspace*
HIDE	: Usado para ocultar um ou mais traçados
DELETE	: Usado para apagar um ou mais traçados
PERSPECTIVE	: Usado para mudar de perspectiva (visão)
RESET	: Usado para reset dos eixos do traçado
HELP	: Exibe a documentação do man Xprism2
QUIT	: Saída do Xprism2

4.Warpimage: É um aplicativo que permite deformação iterativa de imagens.

Outros Aplicativos:

Além do Khoros, existem pacotes menores que permitem o processamento de dados. Es opção tem como objetivo fornecer para o usuário através de um menu novos pacot de processamento (Fig. 3.13). A idéia é de que conforme forem aparecendo aplicativ compatíveis ao ambiente OpenWindows, eles sejam anexados a este menu. Utilizou- como opções o: Xview, XV e o Ftptool.

1.XV: Esta opção permite que se chame o XV. O XV é uma ferramenta de domín público ("XV by John Bradley - V.3.00 Copyright 1993") formado por um ambiente janelas com variadas opções de processamento e que permite visualização de imagens sistema X windows. Usa os formatos de arquivos: GIF, PM, PBM, PGM, PPM, X bitmap, Utah Raster Toolkit RLE, PDS/VICAR, Sun Rasterfile Sun, BMP, PostScrip IRIS, RGB, JPEG e TIFF, em estações de trabalho ou terminais com o sistema X windo V.11.

2.XView:Esta opção do menu permite que se use o XView. O XView Manipula imager de formato .csun, .msun, .sun, .face, .xbm, .bm e .gif. Ao se pressionar o botão SELEC do "mouse" sobre esta opção aparecerá uma janela "popup window" que servirá com base de entrada para a operação.

Forma de uso: xview [opções globais] { [opções para a imagem] nome-da-imagem... }.
Opções globais:

Figura 3.13: Janela do SDVI para Processamento - aplicativos

-onroot	:Carrega a imagem na janela raiz
-boarder colorname	: Cria uma borda na imagem com a cor mencionada
-display dispname	: Destino de visualização
-fullscreen	: Usa a tela inteira para visualização
-geometry WxH+X+Y	: Tamanho do destino e localização
-help	: Mensagem de ajuda
-identify	: Identifica a imagem
-list	: Lista a imagem em seus respectivos diretórios
-install	: Instalação explicita de mapas de cor
-path	: Mostra o caminho para carregar uma imagem
- quiet	: "Silence is golden"
-slideshow	: Mostra a imagem na forma de slide
-supported	: Mostra tipo de imagens que o aplicativo suporta
-verbose	: "Whistle while you work"
-version	: Mostra a versão
-view	: Visualiza uma imagem na tela

Opções de imagem:

-at X,Y	:Carrega a imagem na localização X,Y
-background colorname	: Cor de fundo
-brighten percentage	: Multiplicador de luminância
-center	: Centraliza a imagem
-color number	: Especifica número máximo de cores
-clip X,Y,W,H	: Usar determinada porção da imagem
-dither	: Realiza operação de dither na imagem do tipo bitmap
-foreground colorname	: Cores de frente para a imagem do tipo bitmaps
-name name	: Força o próximo argumento a ser o nome de uma imagem
-xzoom percentage	: Aumenta o eixo X de uma percentagem
-yzoom percentage	: Aumenta o eixo Y de uma percentagem
-zoom percentage	: Aumenta a imagem de uma percentagem

3.FtpTool: Esta opção permite que se use o Ftptool. O ftptool é uma interface em janelas para o ftp. Esconde a iteração com o ftp e a necessidade de comandos conhecidos como 'get' e 'put' e, as diferenças entre 'get' e 'mget'. Prové a abilidade de transferir diretórios, o que o ftp por si só não o faz. Esta opção é viavel devido a necessidade de troca de informações com outros centros (ftp.unicamp.br 'OBELIX', ANONYMOUS), tranferência de arquivos.

Uso do SDVI:

Nesta opção existem funções específicas para os sinais PAL-M e NTSC Fig. 3.14.

O principal objetivo é criar funções de tratamento de sinais de vídeo como: Conversão de freqüências, conversão de formatos, conversão de padrões e, outras funções que permitem a compatibilização de sinais de vídeo.

Conversão de formatos:

Esta opção permite que sejam feitas operações de conversão de formatos, como:

- GIF para VIFF
- VIFF para GIF
- VIFF para Jpeg
- Jpeg para VIFF
- BYTE para VIFF
- VIFF para BYTE

Inicialmente, se sugere o uso do Khoros e/ou o Xv para esse tipo de conversão.

Conversão de sinais:

Existem várias funções de conversão necessárias ao tratamento do sinal de vídeo, em suas respectivas freqüências de operação. Foram adicionados ao sistema vários botões que permitem:

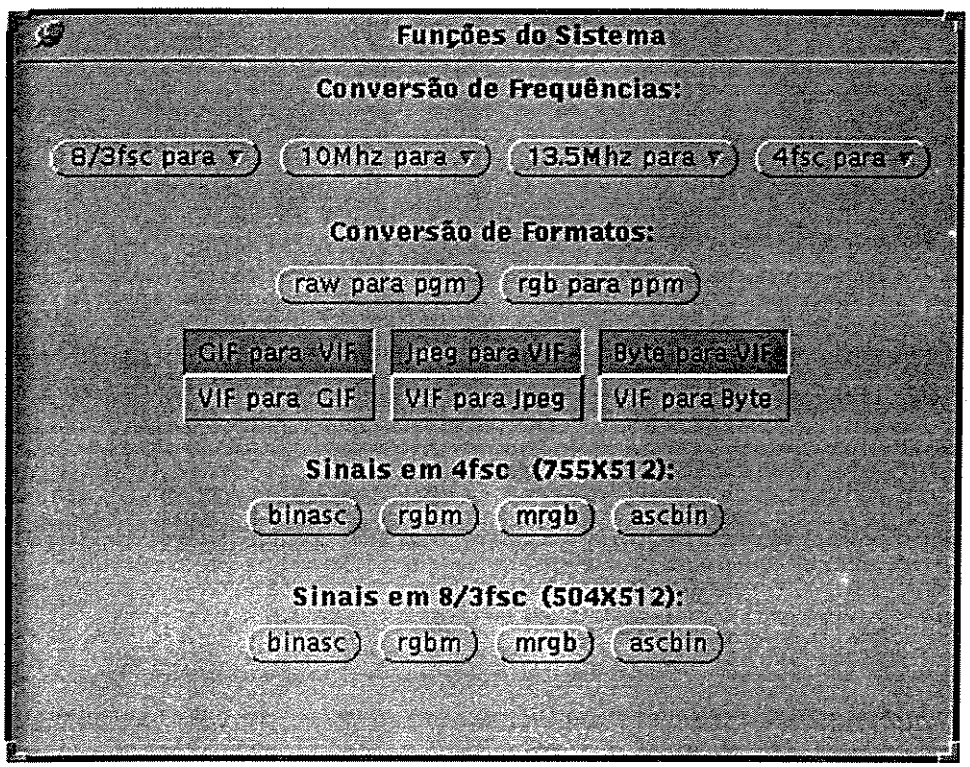

Figura 3.14 Janela de Funções do Sistema.

- Botão raw para pgm: Permite que se transforme um sinal do tipo raw para pgm. Os arquivos de imagem R, G e B de extensão .DAT, são constituídos de símbolos e plenamente crús ou seja sem cabecário e rodapé. São transformados para arquivos de imagens R, G e B com extensão .pgm, (com cabeçário e rodapé) capazes de serem visualizados em níveis de cinza.

Ao se pressionar o botão SELECT do "mouse" sobre essa opção, abre-se uma janela, Fig. 3.15 , que solicita pelos arquivos R, G e B (.DAT) e os converte para R, G e B (.pgm).

- Botão rgb para ppm: Permite que se tranformem os sinais R, G e B (.pgm) para um único sinal composto de extensão .ppm. Os arquivos de imagem R, G e B (.pgm) podem ser visualizados em seus respectivos níveis de cinza, pois já estão em seus formatos de imagem. A transformação para o sinal nome.ppm permite que se visualize o sinal composto em cores.

Ao se pressionar o botão SELECT do "mouse" sobre essa opção, abre-se uma janela, Fig. 3.16 , que solicita pelos arquivos R, G e B (.pgm) e os converte para o composto nome.ppm.

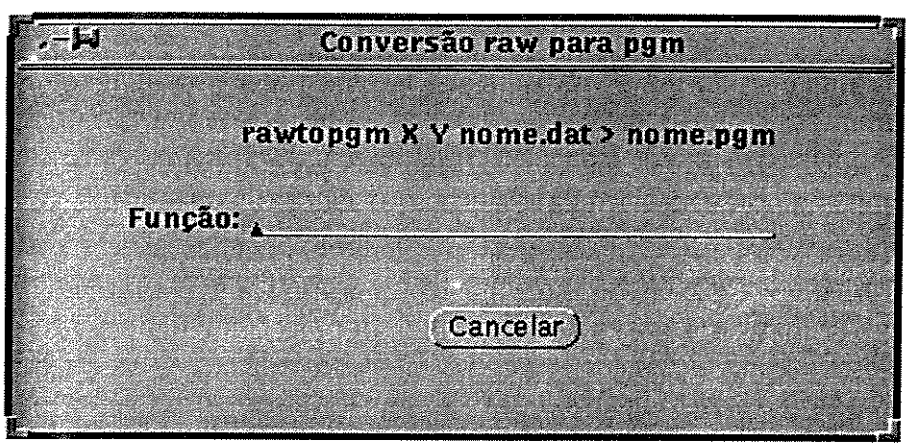

Figura 3.15: Conversão raw para pgm

Há ainda uma subdivisão da janela que permite operar somente com sinais em 4fsc e outra com 8/3fsc. Seus botões respectivos permitem transformações especiais que passamos a descrever:

Sinais em 4fsc

- Botão bintoasc: Permite que se leiam dados BYTE a BYTE de uma imagem de tamanho 755X512 gravados no formato BYTE "0 a 127 e -128 a -1" e os converta para o formato inteiro "0 a 255".

Ao se pressionar o botão SELECT do "mouse" sobre essa opção, abre-se uma janela, Fig. 3.17 , que permite entrar com os arquivos em binário (.DAT) e os transformar para arquivos de formato inteiro (.ASC).

- Botão asctobin: Permite que se leia um quadro de tamanho 755X512 de um sinal de teste/imagem SMPTE que foi gravado no formato inteiro "0 a 255" e o converta para o

formato BYTE "0 a 127 e -128 a -1", gravando o resultado em um arquivo de formato BYTE.

Ao se pressionar o botão SELECT do "mouse" sobre essa opção, abre-se uma janela, Fig. 3.20 , que permite entrar com o arquivo do tipo inteiro e o transforma para arquivo do tipo BYTE.

- **Botão rgbm:** Permite a conversão dos sinais em componetes R, G e B para o sinal composto m.

Ao se pressionar o botão SELECT do "mouse" sobre essa opção, abre-se uma janela, Fig. 3.18 , que permite entrar com os sinais R, G e B (.ASC) e o resultado da composição M (.ASC).

- **Botão mrgb:** Permite a decomposição do sinal composto PAL-M, 755X512 amostras por linhas, nas primárias R, G e B. Tem como objectivo a análise subjetiva da imagem.

Ao se pressionar o botão SELECT do "mouse" sobre essa opção, abre-se uma janela, Fig. 3.19 , que permite entrar com o sinal composto M (.ASC) e o nome dos sinais R, G e B (.ASC) resultantes da decomposição.

Sinais em 8/3fsc

- **Botão bintoasc:** Permite que se faça a conversão de um arquivo no formato BYTE para outro no formato inteiro.

Ao se pressionar o botão SELECT do "mouse" sobre essa opção, abre-se uma janela, Fig.1.18 que permite entrar com os arquivos em binário (.DAT) para os transformar em arquivos de formato inteiro (.ASC). Porém a função de conversão ainda não foi inserida.

- **Botão asctobin:** Permite que se faça a conversão de um sinal gravado no formato inteiro para outro de formato BYTE.

Ao se pressionar o botão SELECT do "mouse" sobre essa opção, abre-se uma janela, Fig.1.21, que permite entrar com o arquivo no formato inteiro (.ASC) para o transformar em um arquivo de formato BYTE. Porém a função de conversão ainda não foi inserida.

- **Botão rgbm:** Permite a conversão dos sinais em componetes R, G e B para o sinal composto m.

Ao se pressionar o botão SELECT do "mouse" sobre essa opção, abre-se uma janela, Fig. 1.19, que permite entrar com os sinais R, G e B (.ASC) e o resultado da composição M (.ASC). Porém a função de conversão ainda não foi inserida.

- **Botão mrgb:** Permite a decomposição do sinal composto PAL-M, nas primárias R, G e B. Tem como objectivo a análise subjetiva da imagem.

Ao se pressionar o botão SELECT do "mouse" sobre essa opção, abre-se uma janela, Fig.1.20, que permite entrar com o sinal composto M (.ASC) e o nome dos sinais R, G e B (.ASC) resultantes da decomposição. Porém a função de conversão ainda não foi

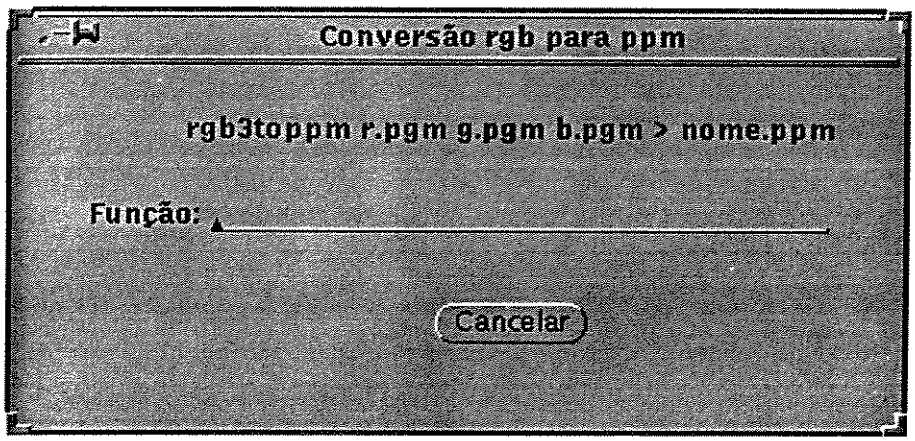

Figura 3.16: Conversão rgb para ppm

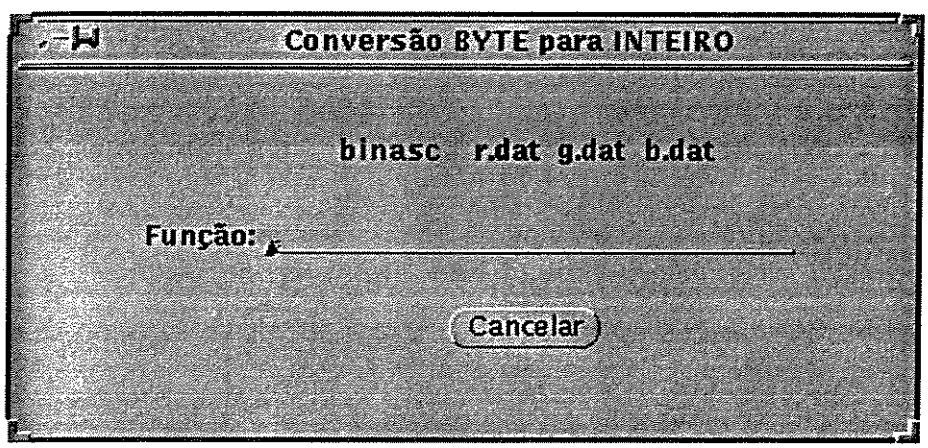

Figura 3.17: Conversão BYTE para INTEIRO

Figura 3.18: Conversão das componentes RGB para o sinal composto M

Figura 3.19: Conversão Composto M para Componentes RGB

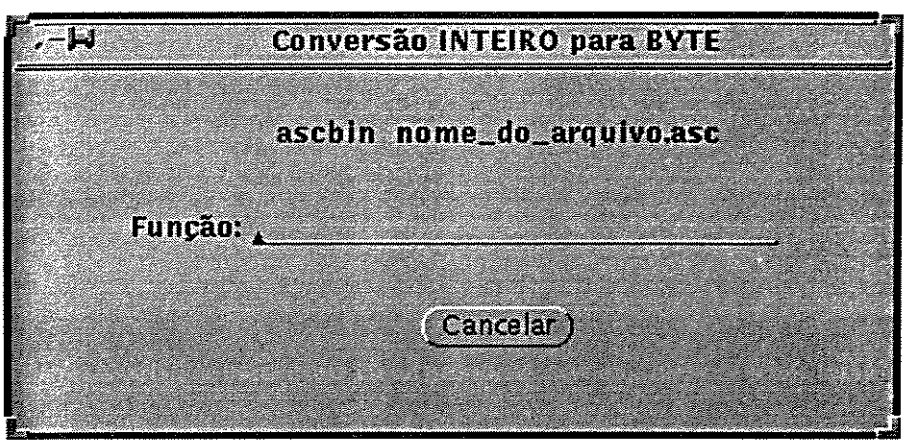

Figura 3.20: Conversão ascii para binário

inserida.

Conversão de freqüências:
Pretende-se trabalhar com sinais de vídeo a uma freqüência de 4fsc, que é a freqüência de operação do digitalizador. Contudo, é viável operações de conversão que permitam gerar sinais de 8/3fsc, 10Mhz e 13.5Mhz de ou para 4fsc, respectivamente. A viabilidade dessas operações de conversão tem como princípio a necessidade do sistema ser compatível com várias freqüências de operação de sinais de vídeo. Assim, decidiu-se criar quatro botões que permitam a conversão entre freqüências ou seja:
- **Botão 8/3fsc para:** Permite que se convertam sinais de vídeo na freqüência de 8/3fsc para: 10Mhz, 13.5Mhz e 4fsc, respectivamente.

Somente a opção 8/3fsc para 4fsc esta sendo implementada. As outras opções serão adicionadas quando implementadas futuramente.
- **Botão 10Mhz para:** Permite que se convertam sinais de vídeo na freqüência de 10Mhz para: 8/3fsc, 13.5Mhz e 4fsc, respectivamente.

Estas operações de conversão serão adicionadas ao sistema quando implementadas futuramente.
- **Botão 13.5Mhz para:** Permite que se convertam sinais de vídeo na freqüência de 13.5Mhz para: 10Mhz, 8/3fsc e 4fsc, respectivamente.

Estas operações de conversão serão adicionadas ao sistema quando implementadas futuramente.
- **Botão 4fsc para:** Permite que se convertam sinais de vídeo na freqüência de 4fsc para: 8/3fsc, 10Mhz e 13.5Mhz, respectivamente.

Estas operações de conversão serão adicionadas ao sistema quando implementadas futuramente.

3.2.4 Visualização

Esta opção tem como objetivo a visualização do sinal de vídeo digitalizado. Sabemos que para o processamento de sinais, existem várias ferramentas de domínio público que permitem visualizar arquivos de imagens em seus diversos formatos. Utilizando-se destas ferramentas, subdividiu-se a opção de visualização, Fig. 3.21, em três alternativas:

1. Uso do Khoros
2. Outros Aplicativos
3. Funções do Sistema

Uso do Khoros:
Para a visualização de imagens usando-se o Khoros, criou-se um menu de cinco opções que permite usar: O Cantata, Putimage, Editimage, Iconimage e Viewimage, Fig. 3.22.

Figura 3.21: Janela do SDVI para Visualização

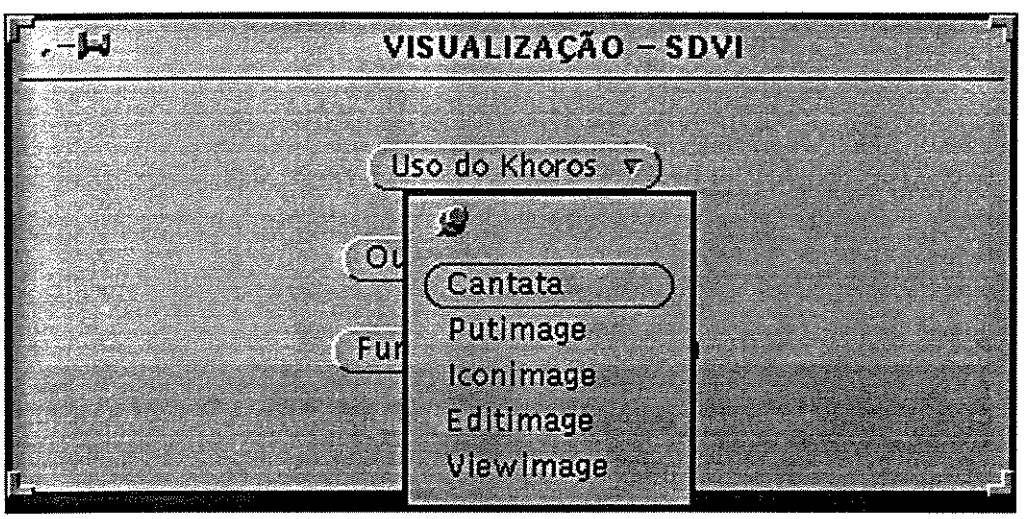

Figura 3.22: Janela do SDVI para Visualização - Uso do Khoros

1.Cantata: Esta opção chama o Cantata. As principais características já foram apresentadas. Internamente por meio de *glyphs*, todas as outras opções podem ser realizadas.

2.Putimage: Esta opção permite que se faça a exibição de uma imagem. Putimage é programa não iterativo que simplesmente exibe imagens de formato .VIFF. A operação é realizada através de um argumento [-i], seguido do nome da imagem que será exibida. Caso a imagem escolhida seja do tipo *multiband*, o programa putimage somente exibirá a primeira. Pressionando o botão SELECT do "mouse" no interior da imagem, fará com que essa desapareça dando fim ao programa.

A sintaxe mínima para putimage é: putimage -i { imagem-de-entrada }.

Contudo, existem argumentos opcionais e que passamos a descrever:

-ov { nome da imagem para sobreposição (overlay) }: Causa a sobreposição deste arquivo de imagem (tipo Bit) com a imagem de entrada. Se a imagem .VIFF passada não for do tipo Bit, será feito um processamento interno para conversão para Bit.

-s { nome de arquivo que serve como máscara }: Com essa opção passa-se um arquivo que serve como máscara para a exibição da imagem de entrada. Aparecem da imagem somente os pontos de valor (1) da máscara os restantes desaparecem.

-c { nome de arquivo máscara }: Neste caso aparecerá a imagem de entrada somente nos pontos cujos pixels da máscara tiverem valor (1), todos outros pontos terão a cor de fundo da janela.
-display display: Pode-se executar putimage de uma workstation e exibir em outra.

3.Iconimage: Esta opção permite que se realize a operação iconimage. Iconimage é um programa não iterativo que permite a exibição de figuras em miniatura, ícones, de imagens de formato .VIFF do Khoros. A imagem é sub-amostrada de acordo com a largura e altura e exibida. Pressionando-se o botão SELECT do "mouse" no interior da imagem esta desaparecerá e o programa iconimage termina.

A sintaxe mínima para iconimage é: iconimage -i { imagem de entrada }
Os argumentos opcionais são:

4.Editimage: Essa opção permite que se chame o aplicativo editimage. Editimage é uma ferramenta que exibe, examina e manipula imagens. Aceita qualquer tipo de imagem .VIFF como entrada; usa imagens .VIFF com dados tipo Bit e Byte, contudo, converte os tipos de dados quando necessário. A imagem de saída (exibida) é do tipo Byte. Correntemente, editimage pode utilizar um máximo de 256 cores. Qualquer imagem que tiver mais de 256 cores é convertida, de forma que suas cores se enquadrem nas 256. Note que esta conversão vai normalizar a imagem com relação ao número total de cores possíveis de serem exibidas na estação de trabalho em uso [16]. Editimage provê ainda várias 'utilidades' para a exibição de imagens. A maior parte destas é com relação aos espaços de cor.

Esta ferramenta foi escrita assumindo que o usuário tenha conhecimento necessário para que corretamente interprete a imagem resultante exibida e, reconheça quando esses resul-

tados não são válidos. Resultados invalidos podem ocorrer quando o espaço de cores da imagem não casa com o espaço de cores do monitor (display).

A janela principal de editimage tem dois menus: "DISPLAY UTILITIES" e "IMAGE UTILITIES".

DISPLAY UTILITIES- Permite acessar: a propriedade de Zoom, os valores dos Pixels, o mapa de cores (LUT, Look Up Table), Pseudo cor, opções de Janelamento e Threshold, de onde o usuário pode obter facilidades. O usuário pode também trocar de imagem (caso de imagens multi-banda), ou sobrepor bits de imagens, criar anotações, salvar e usar o "help" para informações mais detalhadas sobre essas propriedades.

IMAGE UTILITIES- O segundo menu, "IMAGE UTILITIES", provê acesso para a saída/entrada de arquivos, conversão de mapa de cores, região de interesse e modificar as características do cabeçalho da imagem. Podem ser criadas também anotações. Contém a opção "help" que permite o auxílio das propriedades aqui apresentadas.

5.Viewimage: Esta opção permite que se use o viewimage. É um pacote iterativo que permite a exibição da Imagem/Elevação (Imagery/Elevation); É um aplicativo novo que combina os pacotes Xprism3 e Editimage. O objetivo principal de viewimage é de tomar um arquivo de imagem .VIFF do Khoros e usar os seus dados como da Imagem. Esses dados fornecem a "pintura" que é exibida, ajustada em cima dos dados de Elevação que são provinientes de outro arquivo .VIFF do Khoros. É muito importante que o arquivo .VIFF usado como fonte dos dados de Imagem seja do mesmo tamanho que o arquivo .VIFF usado para dados de Elevação. Isto é ter o mesmo número de linhas e colunas.

O viewimage aceita qualquer arquivo .VIFF do Khoros como entrada (Imagery input) e converterá o tipo de imagem quando necessário. Atualmente viewimage aceita um máximo de 256 cores. Qualquer imagem que tiver mais do que 256 cores será convertida automaticamente para outra que se adapte as 256 cores via programa **vgamut**. Note que esta imagem será normalizada para as 256 cores possíveis de se reproduzir na estação de trabalho que está sendo usada.

Outros Aplicativos:

Esta opção permite que se use para a visualização de imagens o Xv ou o XView. Pressionando-se o botão "Outros Aplicativos", aparecerá o menu com as opções: XV e XView (Fig. 3.23). Outros aplicativos de visualização compatíveis com o X windows poderam ser acrescentados conforme forem adquiridos.

1.XV: Esta opção permite que se use o XV. O XV é uma ferramenta iterativa de domínio público ("Desenvolvida por John Bradley - V.3.00 Copyright 1993") formado por um ambiente de janelas com variadas opções de processamento e que permite visualização de imagens no sistema X windows. Usa os formatos de arquivos: GIF, PM, PBM, PGM, PPM, X11 bitmap, BMP, PostScript, IRIS, RGB, JPEG e TIFF, em estações de trabalho ou terminais com o sistema X window V.11.

2.XView: Esta opção do menu permite que se use o xview. O XView Manipula imagens de formato .csun, .msun, .sun, .face, .xbm, .bm e .gif. Ao se pressionar o botão SELECT do "mouse" sobre esta opção aparecerá uma janela "popup window" que servirá como base de entrada para a operação.
Forma de uso: xview [opções globais] { [opções para a imagem] nome-da-imagem }.
As opções globais e de imagem foram descritas anteriormente, veja Processamento-Outros Aplicativos-XView.

Uso do SDVI:
Esse item não tem nenhuma função anexada. Os recursos disponíveis para visualizasão de imagens existentes, satisfazem a necessidade do sistema. É anexada esta opção, devido a maleabilidade que se quer criar. Futuramente, poder-se-á anexar alguma função proviniênte de novas técnicas de visualização.

3.2.5 Informações - Banco de Dados

A implementação de um menu de informações dá ao sistema - SDVI uma característica muito importante, a facilidade de obter informações sobre sistemas de vídeo e trabalhos afins. Essa característica permite a alta concentração de trabalhos que iterativamente o usuário acessa e amplia formando um banco de dados atualizado necessário ao sistema. Atualmente o menu de informações tem as seguintes opções (Fig. 3.24)

1. **O sistema SDVI:**
   Permite obter informações sobre o SDVI. Suas origens, necessidades, interesses do trabalho, composição dos blocos do sistema, características, perspectivas, etc, ...

2. **Laboratório**
   A idéia é que se crie um banco de dados sobre a estrutura física do laboratório; inclui-se nessa opção ferramentas, fontes, geradores de sinais, câmeras de vídeo, micro-computadores, estações de trabalho, equipamentos em montagem e, se possível ainda um "Layout" do espaço físico com a respectiva disposição dos equipamentos. A estrutura é maleável de acordo com aquisições, doações, etc.

3. **Teses e trabalhos publicados:**
   O grupo de vídeo tem a necessidade de concentrar informações sobre teses e trabalhos publicados. Em arquivos PostScript, teremos então a facilidade de adicionar trabalhos ao ambiente de forma que se atualizem dados concernentes ao grupo de vídeo.

4. **Sistema NTSC:**
   Arquivo com informações sobre o sistema em cores NTSC.

5. **Sistema PAL-M:**
   Arquivo com informações sobre o sistema em cores PAL-M.

6. **HDTV:**
   Arquivo com informações sobre o sistema de televisão HDTV.

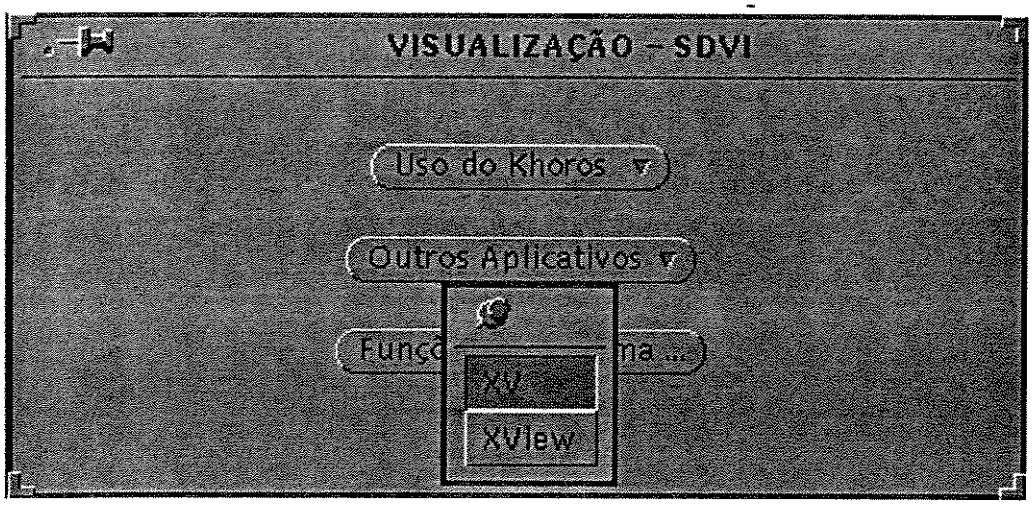

Figura 3.23: Janela do SDVI para Visualização - Outros Aplicativos

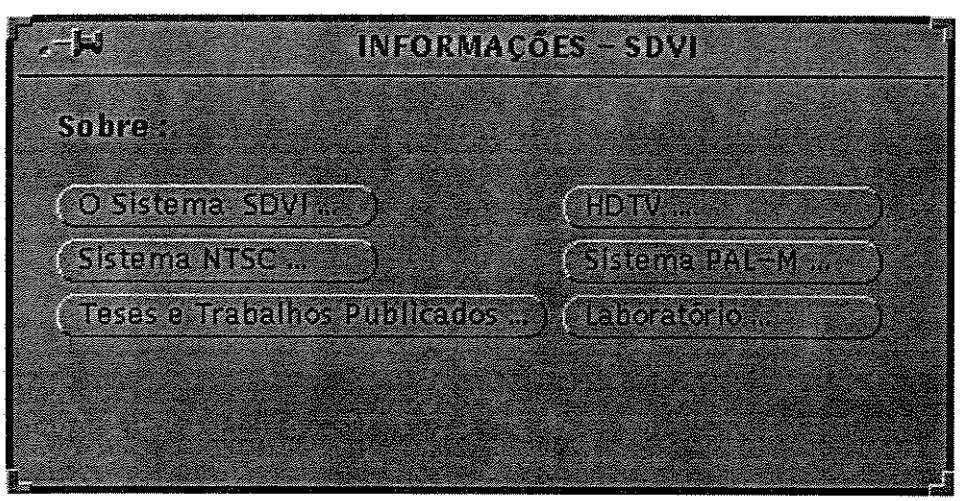

Figura 3.24: Janela do SDVI para Informações - Banco de dados

Todos os arquivos são atualizados conforme mudanças existentes. É possível a criação de opções não existentes no menu.

3.2.6 Maleabilidade do Sistema

O sistema elaborado tem como objetivo a digitalização de sinais de vídeo, armazenamento, processamento, visualização e criação de um banco de dados para o grupo de vídeo. Com essas finalidades o sistema é criado de uma forma que se adapte ao estado de arte, com opções que apesar da não existência das suas funções específicas, possibilitem o acréscimo quando do desenvolvimento de novas técnicas de tratamento de sinais. A adaptação do ambiente ao estado de arte é uma característica importante do sistema.

3.3 Conclusões

Dessa forma as partes desenvolvidas neste trabalho dão suporte e apoio para a viabilização de um sistema de aquisição de imagens que permite:

- a) Geração de banco de dados de imagens. A grande variedade de tipos de imagens correspondentes a cenas estáticas ou dinâmicas requer em muitos casos a escolha de imagens com características típicas que realcem os aspectos ou detalhes desejados para algum estudo específico ;
- b) Obtenção de sinais fonte para avaliação de esquemas de redução de redundância de sinais de vídeo. As ferramentas de trabalho que podem ser desenvolvidos permitirão uma análise mais aprofundada do comportamento dos esquemas propostos;
- c) Avaliação subjetiva da qualidade da imagem. Uma vez que as informações estejam disponíveis num barramento, o sistema possibilita a visualização da imagem correspondente. Controles adicionais podem ser implementados para facilidade de manipulação do sistema.

  Dessa forma, o trabalho aqui elaborado dá suporte e apoio a diversos temas de pesquisa nas áreas de processamento e codificação digitais de imagem, permetindo um melhor aproveitamento dos recursos e organização para aumentar a produção técnico-científica do grupo de trabalho dedicado ao processamento de imagens.

Capítulo 4

Processamento de Sinais de Vídeo. Simulação e resultado de programas de conversão de freqüência

4.1 Introdução

No capítulo 3 falamos sobre o Sistema Digital de Visualização de Imagens. Foi exposta sua funcionalidade e características que lhe são peculiares. Observou-se que o sistema absorve uma série de funções de processamento de sinais provenientes de aplicativos de domínio público, como por exemplo o Khoros que, possui uma estrutura de suporte ao processamento de sinais bastante enriquecida e ampliável segundo suas propriedades de programação.

Ao longo do capítulo 3 pudemos verificar a existência de várias funções desenvolvidas em trabalhos de pesquisa pelo grupo de vídeo.

Este capítulo tem como objetivo adicionar às funções já existentes, funções de conversão de freqüência usando o teorema da amostragem [3].

É feito um breve resumo da teoria de amostragem [3] do qual resulta a expressão do sinal instântaneo em função dos componentes discretos do sinal original (Eq.4.9). Através dos tempos de amostragem os sinais de 4fsc, 8/3fsc e de 13.5Mhz são mapeados em número de amostras ativas e de apagamento horizontal. Esse mapeamento tem como objetivo a configuração dos mosaicos para delinear a programação das funções de conversão. Por meio de simulações, aproximamos a Eq.4.9 por expressões limitadas em componentes discretas (Eqs.4.28, 4.33 e 4.34); para tal foi necessário uma transformação nos coeficientes da função Sample de forma que a soma de seus coeficientes permanecesse constante e igual a um.

Feita a simulação dos programas de conversão de 4fsc para 8/3fsc e de 8/3fsc para 4fsc de sinais em componentes RGB, são apresentados através de tabelas (Tabs.4.3 e 4.5), a relação sinal ruído de vários sinais de teste e, para o teste subjetivo são comparadas e apresentadas imagens (Figs. 4.4, 4.5, 4.7, 4.8, 4.9 e 4.10) que, são avaliadas por vários observadores conforme sugere o CCIR (Tabs. 4.4 e 4.6).

4.2 O Teorema de Amostragem

O teorema de amostragem tem uma grande importância para a teoria da comunicação. Seu enunciado é o seguinte [3]:

Um sinal *f(t)* limitado em faixa, que não tem nenhuma componente espectral acima da freqüência f_mHz, é determinado univocamente por seus valores tomados a intervalos uniformes menores do que $\frac{1}{2f_m}$ segundos [1]. Isso implica que, se a transformada de Fourier de *f(t)* é zero acima de uma certa freqüência $\omega_m = 2\pi f_m$, então a informação completa sobre *f(t)* está contida em suas amostras, espaçadas uniformemente, de uma distância menor do que $\frac{1}{2f_m}$ segundos. Deduz-se, pelo teorema da amostragem, que essas amostras contêm a informação sobre *f(t)* em qualquer valor de *t*. Entretanto a taxa de amostragem deve ser pelo menos duas vezes a freqüência mais alta f_m presente no espectro de *f(t)*. Dizendo de outro modo, o sinal deve ser amostrado pelo menos duas vezes durante cada período ou ciclo de sua componente de freqüência mais alta.

4.2.1 Recuperação de *f(t)* a partir de suas amostras

Considere o sinal limitado em faixa *f(t)*, que não tem nenhuma componente espectral acima de f_m ciclos por segundo. Isso significa que $F(\omega)$, a transformada de Fourier de *f(t)*, é zero para $|\omega| > \omega_m (\omega_m = 2\pi f_m)$. Suponha que multipliquemos a função *f(t)* por uma função impulso periódica $\delta_T(t)$. A função produto é uma seqüência de impulsos localizados a intervalos regulares de T segundos e com pesos iguais aos valores de *f(t)* nos instantes correspondentes. O produto $f(t)\delta_T(t)$ representa, na verdade, a função *f(t)* amostrada a intervalos uniformes de T segundos. Denotaremos essa função amostrada por $f_s(t)$.

$$f_s(t) = f(t)\delta_T(t)$$

O espectro de freqüência de *f(t)* é $F(\omega)$. A transformada de Fourier de um trem uniforme de funções impulso δ_t é também um trem uniforme de funções impulso $\omega_0\delta_{\omega_0}(\omega)$. Os impulsos são separados por um intervalo uniforme $\omega_0 = \frac{2\pi}{T}$.

$$\delta_T(t) \leftrightarrow \omega_0\delta_{\omega_0}(\omega)$$

A transformada de Fourier de $f(t)\delta_T(t)$ será dada de acordo com o teorema da convolução em freqüência, pela convolução de $F(\omega)$ com $\omega_0\delta_{\omega_0}(\omega)$.

---

[1] Esse teorema é, na realidade, um caso especial do teorema geral de amostragem, cujo enunciado é o seguinte: Se um sinal é limitado em faixa, e se o intervalo de tempo é dividido em partes iguais, formando intervalos tais que, cada subdivisão compreenda um intervalo de duração T segundos, onde T é menor do que $\frac{1}{2f_m}$, e se uma amostra instantânea é tomada arbitrariamente de cada subintervalo, então o conhecimento da magnitude instantânia de cada amostra mais o conhecimento dos instantes em que é tomada a amostra de cada subintervalo contêm toda a informação do sinal original.

Veja , por exemplo, H.S. Black, Modulation Theory, Van Nostrand, New York, 1953, p.41 .

$$f_s(t) \leftrightarrow \frac{1}{2\pi}[F(\omega) * \omega_0 \delta_{\omega_0}(\omega)]$$

Substituindo $\omega_0 = \frac{2\pi}{T}$, obtemos

$$f_s(t) \leftrightarrow \frac{1}{T}[F(\omega) * \delta_{\omega_0}(\omega)] \tag{4.1}$$

Pela Eq.4.1, é evidente que o espectro do sinal amostrado $f_s(t)$ é dado pela convolução de $F(\omega)$ com um trem de impulsos. A função de densidade espectral (Transformada de Fourier) de $f_s(t)$ é, portanto, a mesma função $F(\omega)$, mas que se repete periodicamente a cada ω_0 radianos por segundo. Essa função será chamada $F_s(\omega)$. Observe que $F(\omega)$ se repetirá periodicamente, sem superposição, uma vez que $\omega_0 \geq 2\omega_m$, ou

$$\frac{2\pi}{T} \geq 2(2\pi f_m)$$

ou seja,

$$T \leq \frac{1}{2f_m} \tag{4.2}$$

O intervalo máximo de amostragem $T = \frac{1}{2f_m}$ também é chamado de *intervalo de Nyquist.*

Sabendo que:

$$\begin{aligned} \delta_{\omega_0} &= \delta(\omega) + \delta(\omega - \omega_0) + \cdots + \delta(\omega - n\omega_0) + \cdots \\ &\quad + \delta(\omega + \omega_0) + \cdots + \delta(\omega + n\omega_0) + \cdots \\ &= \textstyle\sum_{n=-\infty}^{\infty} \delta(\omega - n\omega_0) \end{aligned}$$

Pela Eq.4.1, segue-se que:

$$\begin{aligned} F_s(\omega) &= \tfrac{1}{T}[F(\omega) * \delta_{\omega_0}(\omega)] \\ &= \tfrac{1}{T}\left[F(\omega) * \textstyle\sum_{n=-\infty}^{\infty} \delta(\omega - n\omega_0)\right] \\ &= \tfrac{1}{T} \textstyle\sum_{n=-\infty}^{\infty} F(\omega) * \delta(\omega - n\omega_0) \end{aligned}$$

Por outra, usando-se a Eq.4.3,

$$f(t) * \delta(t - T) = f(t - T) \tag{4.3}$$

tem-se,

$$F_s(\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} F(\omega - n\omega_0) \tag{4.4}$$

A função original pode ser recuperada, passando-se a função amostrada através de um filtro passa-baixas com uma freqüência de corte ω_m (Fig.4.1). Evidentemente, essa é uma operação no domínio da freqüência. Devido à dualidade entre o domínio da freqüência e o domínio do tempo, há uma operação equivalente no domínio do tempo para se recuperar *f(t)* a partir das suas amostras.

Consideremos um sinal *f(t)* amostrado a uma taxa mínima necessária ($2f_m$ amostras por segundo). Nesse caso,

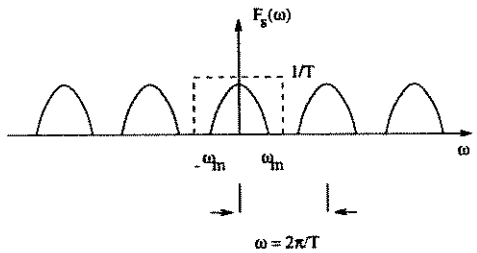

Figura 4.1: Características do filtro passa-baixas.

$$T = \frac{1}{2f_m} \quad e \quad \omega_0 = \frac{2\pi}{T} = 4\pi f_m = 2\omega_m$$

Logo, a Eq.4.4 torna-se

$$F_s(\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} F(\omega - 2n\omega_m) \tag{4.5}$$

Como já se observou, o espectro $F(\omega)$ pode ser obtido filtrando-se $F_s(\omega)$ através de um filtro passa-baixas de freqüência de corte ω_m. É evidente que essa operação de filtragem é equivalente à multiplicação de $F_s(\omega)$ por uma função porta $G_{2\omega_m}(\omega)$. Logo pela Eq.4.5, obtemos

$$F_s(\omega)G_{2\omega m}(\omega) = \frac{1}{T}F(\omega)$$

Portanto,

$$F(\omega) = TF_s(\omega)G_{2\omega m}(\omega) \tag{4.6}$$

Assim, a transmissão do sinal amostrado $f_s(t)$ através de um filtro passa-baixas resulta no sinal $f(t)$. O filtro tem uma freqüência de corte ω_m e um ganho de $T = \frac{1}{2f_m}$. A função de

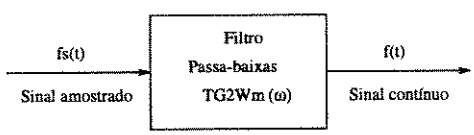

Figura 4.2: Filtro Passa-baixas.

transferência $H(\omega)$ desse filtro (Fig.4.2) pode ser expressa como:

$$\begin{aligned} H(\omega) &= TG_{2\omega m}(\omega) \\ &= \frac{1}{2f_m}G_{2\omega m}(\omega) \end{aligned}$$

A aplicação do teorema da convolução no tempo à Eq.4.6 resulta em

$$\begin{aligned} f(t) &= Tf_s(t) * \frac{\omega_m}{\pi} Sa(\omega_m t) \\ &= f_s(t) * Sa(\omega_m t) \end{aligned} \tag{4.7}$$

A função amostrada $f_s(t)$ é dada por:

$$f_s(t) = \sum_{n=-\infty}^{\infty} f_n \delta(t - nT)$$

onde f_n é a n-ésima amostra de *f(t)*. Logo,

$$\begin{aligned} f(t) &= \sum_{n=-\infty}^{\infty} f_n \delta(t - nT) * Sa(\omega_m t) \\ &= \sum_{n=-\infty}^{\infty} f_n Sa[\omega_m(t - nT)] \qquad (4.8) \\ &= \sum_{n=-\infty}^{\infty} f_n Sa(\omega_m t - n\pi) \qquad (4.9) \end{aligned}$$

É evidente que *f(t)* pode ser construída no domínio do tempo a partir de suas amostras, de acordo com a Eq.4.9. Cada amostra é multiplicada por uma função de amostragem e todas as formas de onda resultantes são somadas para se obter *f(t)*.

Na prática, a maior parte dos sinais se aproxima bastante dos sinais limitados em faixa. Convém esclarecer que, rigorosamente falando, um sinal limitado em faixa não existe. Entretanto, na prática, para todos os sinais, as funções de densidade espectral diminuem nas freqüências superiores. A maior parte de energia é levada pelas componentes que ocupam um certo intervalo de freqüência e, para todos os propósitos práticos, um sinal pode ser considerado limitado em faixa. O erro, que resulta por não se levar em conta as componentes de alta freqüência, é desprezível.

O teorema de amostragem é um conceito importante, pois nos permite substituir um sinal contínuo limitado em faixa por uma seqüência discreta de suas amostras, sem perda de qualquer informação. Assim, o conteúdo de informação de um sinal contínuo limitado em faixa é equivalente a partes discretas de informação. Uma vez que o princípio de amostragem especifica o número mínimo de valores discretos necessários, em um determinado intervalo de tempo, para se reproduzir um sinal contínuo, o problema da transmissão desse sinal se reduz ao da transmissão de um número finito de valores. Essa informação discreta pode ser transmitida por um grupo de pulsos, cujas amplitudes variam de acordo com os valores das amostras (modulação de amplitude de pulso). Outras formas de modulação são a modulação de posição de pulso, onde a posição do pulso varia, a modulação de largura de pulso (a variação da largura do pulso é proporcional aos valores da amostra), ou modulação por código de pulso (onde as amostras são representadas por um código formado por um grupo de pulsos).

4.2.2 Teorema de amostragem (domínio da freqüência)

O teorema da amostragem no domínio do tempo tem um dual cujo enunciado é:

Um sinal limitado em tempo que vale zero para $|t| \rangle T$ é determinado univocamente pelas amostras do seu espectro de freqüência, tomadas a intervalos uniformes menores que $\frac{1}{2T}$ Hz (ou $\frac{\pi}{T}$ radianos por segundo), ou seja [3]:

$$F(\omega) = \sum_{n=-\infty}^{\infty} F\left(\frac{n\pi}{T}\right) Sa(\omega T - n\pi) \tag{4.10}$$

4.2.3 Uma aproximação para o sinal instantâneo $f(t_0)$

Considerando a Eq. 4.9, percebemos que um valor instantâneo $f(t_0)$ é resultado da soma de uma infinidade de harmônicas, influência de n amostras com peso f_n, que se sobrepõem em determinado instante. Conclui-se também que, conforme se afasta a n-ésima amostra do ponto analisado, esta terá menor influência no resultado da amplitude do instante t_0.

Dessa forma, para diminuir o tempo de processamento, já que uma imagem possui um elevado número de pontos à processar, foram feitos vários testes, considerando a influência do afastamento da n-ésima amostra do instante t_0 em análise. A Eq.4.9, passou a ser analisada como sendo uma aproximação, fazendo para isso $|n| \leq N$, ou seja:

$$f(t) = \sum_{n=-N}^{N+1} f_n Sa(\omega_m t - n\pi) \qquad N \text{ inteiro.} \tag{4.11}$$

Percebe-se ainda da Eq. 4.9 que se $f(t) = K$ (K: Constante qualquer), então o peso da n-ésima amostra f_n para o instante analisado é K, ou seja:

$$\begin{aligned} f(t) &= \sum_{n=-\infty}^{\infty} f_n Sa(\omega_m t - n\pi) \text{ para } t = t_0 \text{ e } f_n = K \\ f(t_0) &= \sum_{n=-\infty}^{\infty} K Sa(\omega_m t_0 - n\pi) \\ K &= K \sum_{n=-\infty}^{\infty} Sa(\omega_m t_0 - n\pi) \end{aligned}$$

Concluindo-se que:

$$\sum_{n=-\infty}^{\infty} Sa(\omega_m t - n\pi) = 1 \tag{4.12}$$

Para $|n| \leq N$:

$$\sum_{n=-N}^{N+1} Sa(\omega_m t - n\pi) = 1 \tag{4.13}$$

Essa última imposição, provoca a necessidade do acréscimo ou decréscimo ponderado da diferença entre a soma real dos coeficientes e o valor unitário, para cada coeficiente da função Sample, de forma que satisfaça a expressão 4.13.

Além disso é importante que a amostra a ser recuperada esteja sempre aproximadamente no centro das amostras usadas. Isso é conseguido fazendo-se um deslocamento contínuo conforme a Eq.4.11.

4.3 O sinal em $4fsc$

O sistema SDVI digitaliza sinais a uma freqüência de 4fsc. A importância da digitalização a essa taxa é dada pela originalidade da informação. Por ser uma taxa super-Nyquist a fidelidade de visualização é muito alta. Essa taxa de amostragem, tem a vantagem das amostras estarem regularmente espaçadas dentro de uma linha, formando um mosaico com amostras alinhadas verticalmente[14]. Essa freqüência tem também a vantagem da facilidade de conversão para todas as outras menores (8/3fsc, 3fsc, 13.5Mhz,...). Como a digitalização é feita para dois sistemas em cores, procuramos apresentar à priori a disposição das amostras conforme mosaico (Fig.4.3) e Tab. 4.1.

Para o sistema em cores NTSC e freqüência de amostragem fa=4fsc, são válidas as seguintes expressões [14]:

$$fa = 4fsc \tag{4.14}$$

$$fsc = \frac{455}{2} f_H \tag{4.15}$$

$$fa = 4\frac{455}{2} f_H \tag{4.16}$$

Da Eq.4.16 temos um total de:

- 910 amostras na linha;
- 455 (=910/2) amostras para os sinais diferença.

Para o sistema em cores PAL-M, valem as expressões:

$$fa = 4fsc \tag{4.17}$$

$$fsc = \frac{909}{4} f_H \tag{4.18}$$

$$fa = 4\frac{909}{4} f_H \tag{4.19}$$

Da Eq.4.19 temos um total de:

- 909 amostras na linha;
- 454/455 (=909/2) amostras para os sinais diferença.

Com o número de amostras da linha, o tempo T_H correspondente e o tempo de Apagamento Horizontal T_{APH} (Fig.II.31 e Tab. II.3 de [15]), é feita através da regra de três o cálculo do número de amostras de Apagamento Horizontal e por consequência das Amostras Ativas da linha para os sistemas PAL-M e NTSC (Tab.4.1).

Disposição das amostras para fa=4fsc							
	bméd.	bmáx.	bmín.	(b+c)máxs.	(b+c)míms	(b+c)méd	Usado
T_{APH} (μs)	9.6	10.3	8.9	12.84	10.17	11.505	
Naa NTSC(910)	772	762	782	726	764	745	755
N_{APH} NTSC	138	148	128	184	146	165	155
Naa PAL-M(909)	771	761	781	725	763	744	755
N_{APH} PAL-M	138	148	128	184	146	165	154

Tabela 4.1: Número de Amostras Ativas(Naa) e de Apagamento Horizontal (APH) em uma linha para fa=4fsc.

4.4 O sinal em $8/3fsc$

A) Sistema em cores NTSC

Para o sistema NTSC com freqüência de amostragem de 8/3fsc, são válidas as seguintes equações [14]:

$$fa = 8/3fsc \tag{4.20}$$

$$fsc = \frac{455}{2} f_H \tag{4.21}$$

$$fa = \frac{8}{3}\frac{455}{2} f_H \tag{4.22}$$

Da Eq.4.22 temos um total de:

- 606/607 amostras na linha. Não existe alinhamento vertical das amostras;
- 303 (=606/2) amostras para os sinais diferença.

Em particular, para o sistema em cores NTSC, o estudo das possíveis freqüências de amostragem: 2fsc, 3fsc,8/3fsc, 4fsc, 12/5fsc, 14fsc; revelam:
- **2fsc:** Tem a desvantagem por ser uma taxa sub-Nyquist por isso não é usada;
- **3fsc e 8/3fsc:** Resultam em amostras não alinhadas, não aconselhável;
- **4fsc, 12/5fsc e 14/5fsc:** Resultam em amostras alinhadas. O ciclo de 12/5fsc e 14/5fsc é muito alto, por isso não é aconselhável;

Logo, resulta que para NTSC, convém que se trabalhe na freqüência de 4fsc.

B) Sistema em cores PAL-M

Para o sistema PAL-M são válidas as equações:

$$fa = 8/3fsc \tag{4.23}$$

$$fsc = \frac{909}{4} f_H \tag{4.24}$$

$$fa = \frac{8}{3}\frac{909}{4} f_H \tag{4.25}$$

Da Eq.4.25 temos um total de:

- 606 amostras na linha;
- 303 (=606/2) amostras para os sinais diferença.

Esse número, 606 amostras na linha e 303 amostras de sinal diferença, gera um mosaico com amostras alinhadas verticalmente.

Tendo o tempo total da linha $T_H = 63,492\mu s$ com o seu respectivo número de amostras (606), considerando o tempo de Apagamento Horizontal (Fig II.31 e Tab. II.3 de [15]) temos através da regra de três os resultados da Tab.4.2

Disposição das amostras para fa=8/3fsc							
N_{AT} 606 T_{APH} (μs)	bméd. 9.6	bmáx. 10.3	bmín. 8.9	(b+c)máxs. 12.84	(b+c)míms 10.17	(b+c)méd 11.505	Usado
Naa	514	507	521	483	509	496	504
N_{APH}	92	99	85	123	97	110	102

Tabela 4.2: Número de Amostras Ativas(Naa) e de Apagamento Horizontal (APH) em uma linha para fa=8/3fsc.

4.5 Conversão da freqüência de $4fsc$ para $8/3fsc$

Para a conversão da frequência de 4fsc para 8/3fsc usou-se a aproximação considerada na Eq.4.11. Recordando, o valor instantâneo $f(t_0)$ (Eq.4.9), é o resultado da soma de uma infinidade de harmônicas que se sobrepõem em determinado instante t_0. Devido a menor influência das harmônicas existente na amplitude do sinal, no instante considerado, a medida que o peso f_n se afasta do ponto em análise e, por consequência para a redução do tempo de processamento, a Eq.4.9

$$f(t) = \sum_{n=-\infty}^{\infty} f_n Sa(\omega_m t - n\pi) \tag{4.26}$$

é simplificada para:

$$f(t) = \sum_{n=-N}^{N+1} f_n Sa(\omega_m t - n\pi) \qquad N \text{ inteiro.} \tag{4.27}$$

satisfazendo a Eq. 4.13.

Por conseguinte, uma comparação entre os mosaicos dos sinais em 4fsc e 8/3fsc (Fig.4.3), mostra a regularidade entre as amostras alinhadas verticalmente e, de acordo com a nova taxa

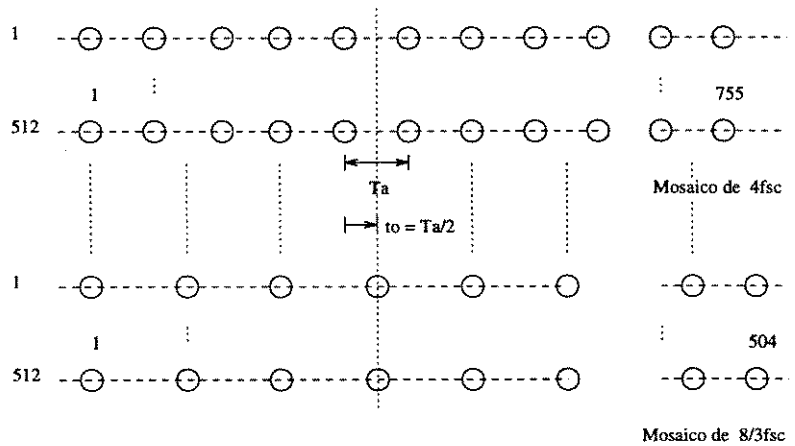

Figura 4.3: Comparação entre os mosaicos de 4fsc e 8/3fsc.

de amostragem, percebe-se pela comparação que, existem instantes de amostragem coincidentes em que são desnecessários os cálculos do sinal na nova frequência através da Eq.4.27. Considerando os instantes de amostragem não coincidentes como sendo $t = t_0 = \frac{T_a}{2}$ (veja Fig.4.3) e, substituindo a frequência de corte $\omega_m = \frac{\omega_a}{2}$, temos da Eq.4.27:

$$\begin{aligned} f(t) &= \sum_{n=-N}^{N+1} f_n Sa(\frac{\omega_a}{2}\frac{T_a}{2} - n\pi) \\ &= \sum_{n=-N}^{N+1} f_n Sa(\frac{2\pi f_a}{2}\frac{T_a}{2} - n\pi) \\ &= \sum_{n=-N}^{N+1} f_n Sa(\frac{\pi}{2} - n\pi) \\ &= \sum_{n=-N}^{N+1} f_n \frac{\sin(\frac{\pi}{2} - n\pi)}{(\frac{\pi}{2} - n\pi)} \quad onde\ t = \frac{T_a}{2} \end{aligned} \tag{4.28}$$

Por analogia, da Eq. 4.13 temos:

$$\sum_{n=-N}^{N+1} \frac{\sin(\frac{\pi}{2} - n\pi)}{(\frac{\pi}{2} - n\pi)} = 1 \tag{4.29}$$

Levando em conta essa informação, de acordo com o valor de N para a aproximação, o programa calcula inicialmente os coeficientes da função $Sa(\frac{\pi}{2} - n\pi)$, acrescentando posteriormente a esses coeficientes a média ponderada no sentido de que se satisfaça a Eq.4.29.

Através de simulações, fazendo variar o valor de N na Eq.4.28, foram analisados vários resultados comparando o sinal calculado pela expressão à valores originais gerados na freqüência

de 4fsc e, optamos por $N = 3$ que leva aos menores erros quando consideramos sinais com muitas variações. O resultado final dos testes de simulação são apresentados na Tab.4.3.

Ao longo do programa, o sinal é testado e se verifica sua semelhança com:

A) **Um sinal constante**, para o qual se determina $N = 0$, ou seja existem 2 amostras para o cálculo do sinal;

B) **Uma rampa** de inclinação mais ou menos constante com uma variação $\Delta = \pm 1$. Para esse sinal usou-se $N = 1$ ou seja, 4 amostras para o cálculo do sinal;

C) **Uma senóide**. Como última opção o sinal se assemelharia a uma senóide (sinal com muita variação), o que para efeito usou-se $N = 3$; um total de 8 amostras para o cálculo do sinal.

Para mais detalhes acerca do programa consultar Apêndice A. O programa esta em linguagem C e é de fácil compreensão devido aos comentários feitos ao longo de suas linhas de instrução.

4.5.1 Simulação e Resultados

Para a simulação foram usados seis sinais de teste e os resultados para a avaliação objetiva são apresentados na tabela 4.3.

Sinal de teste:	SNR [dB]
BARRAS COLORIDAS com amplitude de 100%	45.125027
BARRAS COLORIDAS com amplitude de 75%	47.624310
CCIR-I	49.257481
CCIR-II	45.878323
ONDA TRIÂNGULAR MODULADA	54.260406
ONDA SENOIDAL DE DOIS PERIODOS	57.344948

Tabela 4.3: Resultados da simulação para seis sinais de teste.

Esses resultados foram calculados sem levar em consideração as transições de fase do sinal, onde se verificam erros que, devido a localização são imperceptíveis ao olho humano.

Para a avaliação subjetiva foi usada a imagem COZINHA (SMPTE15) com uma freqüência original de 4fsc (755X512 pels, amostras ativas) e convertida para 8/3fsc (504X512 pels, amostras ativas), veja figs.4.5 e 4.4. Seis observadores avaliaram a semelhança entre as duas imagens e, conforme sugerido pelo CCIR, aplicaram notas que poderiam variar de 1 a 5. O resultado foi o da Tab.4.4:

IMAGEM:	NOTA
Cozinha (SMPTE15)	5

Tabela 4.4: Resultados da avaliação subjetiva.

4.6 Conversão da freqüência de 8/3fsc para 4fsc

Para essa conversão usou-se a aproximação dada pela Eq.4.11, veja Eq.4.32

$$f(t) = \sum_{n=-N}^{N+1} f_n Sa(\omega_m t - n\pi) \quad N \ inteiro \tag{4.30}$$

$$\omega_m = \frac{\omega_a}{2} = \frac{2\pi fa}{2} = \frac{\pi}{T_a} \tag{4.31}$$

$$f(t) = \sum_{n=-N}^{N+1} f_n Sa(\frac{\pi}{T_a} t - n\pi) \quad N \ inteiro \tag{4.32}$$

Consideramos que os dois sinais estão em fase, por isso, a primeira amostra é tida com referência, sendo para isso a mesma em ambas as freqüências. Nesse caso não necessitamos de cálculos através da expressão 4.32. Concluimos também pela comparação entre mosaicos que, além da primeira amostra existe um conjunto de amostras em que os instantes de amostragem são os mesmos para ambas as freqüências. Para esse conjunto de amostras, evitamos cálculos quer pela vantagem da redução do tempo de processamento como também pelos erros de aproximação a que nos sujeitamos usando 4.32 (veja mosaicos na Fig.4.6).

Dando sequência as amostras seguintes, devido a falta de simetria encontrada nos instantes de amostragem ao se compararem os dois mosaicos (4fsc e 8/3fsc), desenvolvemos da Eq. 4.32 duas expressões: uma relacioda com o instante $t = t_1 = \frac{2}{3}T_a$ (Eq.4.33), tendo como referência no caso da Fig.4.6 a primeira amostra do mosaico de 8/3fsc e a segunda expressão, a Eq.4.34 em que consideramos $t = t_2 = \frac{2}{6}T_a$ onde, a amostra de referência para o caso da Fig.4.6 (mosaico de 8/3fsc) é a segunda.

$$f(t) = \sum_{n=-N}^{N+1} f_n Sa(\frac{2}{3}\pi - n\pi) \quad N \ \ inteiro, \ \ t = t_1 = \frac{2}{3}T_a \tag{4.33}$$

$$f(t) = \sum_{n=-N}^{N+1} f_n Sa(\frac{2}{6}\pi - n\pi) \quad N \ \ inteiro, \ \ t = t_2 = \frac{2}{6}T_a \tag{4.34}$$

Devido a redução feita no número de amostras para o cálculo do valor instantâneo da amostra em 4fsc, foi necessário acrescentar aos coeficientes da função Sample, uma porcentagem ponderada para satisfazer como na Eq.4.13 as equações:

$$\sum_{n=-N}^{N+1} \frac{\sin(\frac{2}{3}\pi - n\pi)}{\frac{2}{3}\pi - n\pi} = 1 \tag{4.35}$$

$$\sum_{n=-N}^{N+1} \frac{\sin(\frac{2}{6}\pi - n\pi)}{\frac{2}{6}\pi - n\pi} = 1 \tag{4.36}$$

As Eqs.4.33 e 4.34 satisfazendo 4.35 e 4.36 deslocam-se ao longo das várias linhas da imagem perfazendo o cálculo da imagem processada na freqüência de 4fsc.

Por meio de simulações, fazendo variar o valor de N, otimizou-se o programa para o menor erro possível. Assim, considerando três tipos de sinal conclui-se pelas simulações que, para:

A) **Um sinal constante**, $N = 0$, ou seja o sinal é calculado através de duas amostras sem erro e com baixo tempo de processamento;

B) **Uma rampa** de inclinação mais ou menos constante com uma variação $\Delta = \pm 1$. $N = 1$, ou seja o sinal é calculado através de 4 amostras, número assim reduzido de forma que se evitem transições do tipo das existentes em topos de ondas triângulares onde para o caso, o programa considera essa região do sinal como se pertencesse a uma senóide;

C) **Uma senóide**. Como última opção o sinal se assemelharia a uma senóide, sinal com muita variação, o que para efeito usou-se $N = 2$, um total de 6 amostras para o cálculo do sinal. O valor de N como sendo 2 para esse último caso foi encontrado por meio de testes para uma senóide e por conseguinte para os seis sinais de teste Tab.4.5. Verificou-se um aumento considerável da relação sinal ruído passando o valor de N de 3 para 2.

Para mais detalhes acerca do programa consulte o Apêndice B. O programa esta em linguagem C e é facilmente entendido devido aos comentários feitos ao longo de suas linhas de instrução.

4.6.1 Simulação e Resultados

Para a simulação foram usados novamente seis sinais de teste. Os valores calculados através do programa conforme equações representadas anteriormente, foram comparados com o sinal original, gerado através de software apropriado. O resultado foi o da Tab.4.5.

Sinal de teste:	SNR [dB]
BARRAS COLORIDAS com amplitude de 100%	31.464966
BARRAS COLORIDAS com amplitude de 75%	33.947346
CCIR-I	34.106182
CCIR-II	39.312107
ONDA TRIÂNGULAR MODULADA	56.058281
ONDA SENOIDAL DE DOIS PERIODOS	56.718479

Tabela 4.5: Resultados da simulação objetiva para seis sinais de teste.

Esses valores foram calculados sem levar em consideração as transições de fase do sinal, onde se verificam erros, que devido a localização (mudança de cor) são imperceptíveis ao olho humano.

Para a avaliação subjetiva foram usadas as imagens sala (SMPTE 02) e sala-escura (SMPTE 08) a uma freqüência original de 8/3fsc (504X512 amostras ativas) e convertidas para 4fsc (755X512 amostras ativas), veja Figs.4.7, 4.8, 4.9 e 4.10. As imagens foram avaliadas comparando-as com as originais por seis observadores que, conforme sugerido pelo CCIR aplicaram notas que poderiam variar de 1 a 5. O resultado foi o da Tab.4.6.

IMAGEM:	NOTA
Sala (SMPTE 02)	5
Sala Escura (SMPTE 08)	5

Tabela 4.6: Resultados da avaliação subjetiva.

4.7 O sinal em 13.5 Mhz

A) Sistema em cores NTSC
Considerando as expressões 4.37, 4.38 e 4.39 :

$$fa = 13,5Mhz \tag{4.37}$$

$$fsc = \frac{455}{2} f_H \tag{4.38}$$

$$\frac{fa}{6} = 143 f_H \tag{4.39}$$

pode-se provar por simples substituição da Eq.4.38 na Eq.4.39 ou de 4.39 em 4.38 que, a freqüência de amostragem de 13,5Mhz satisfaz as expressões 4.37, 4.38 e 4.39, respectivamente. Assim, de 4.39:

$$fa = 143 \times 6 \times f_H \tag{4.40}$$

substituindo 4.38 e fsc = 3.5759545 Mhz, temos:

$$\bullet fa = 13.5Mhz$$

Invertendo, ou seja apartir de 4.38:

$$fsc = \frac{455}{2} f_H \tag{4.41}$$

substituindo f_h da Eq.4.39 e fa=13.5Mhz, temos:

$$\bullet fsc = 3,579545455Mhz$$

Portanto, uma vez que as Eqs.4.37, 4.38 e 4.39 satisfazem o sistema em cores NTSC para a freqüência de 13,5Mhz, podemos calcular o número de amostras totais em uma linha usando para isso a expressão 4.39.Da Eq.4.39:

$$T_H = 858T_a \tag{4.42}$$

logo, temos um total de:

- 858 amostras na linha;
- 429 (=858/2) amostras para os sinais diferença.

Considere os tempos de Apagamento Horizontal (Fig.II.31 e Tab.II.3) [15], como sendo os da Tab.4.7.

Sabendo que para o tempo total de uma linha, $T_H = 63,492\mu s$, temos 858 amostras, através dos tempos de APH presumíveis e de uma regra de três simples, chegamos aos valores de N_{APH} - Número de Amostras de Apagamento Horizontal e por consequência a Naa - Número de Amostras Ativas na Linha. Os resultados são transpostos na Tab.4.7.

Disposição das amostras para fa=13.5Mhz							
N_{AT} 858	bméd.	bmáx.	bmín.	(b+c)máxs.	(b+c)míms	(b+c)méd	Para uso
T_{APH} (μs)	9.6	10.3	8.9	12.84	10.17	11.505	10.17
Naa	728	718	737	684	720	702	720
N_{APH}	130	140	121	174	138	156	138

Tabela 4.7: Número de Amostras Ativas(Naa) e de Apagamento Horizontal (APH) em uma linha para fa=13.5Mhz.

B) Sistema em cores PAL-M

Dadas as seguintes equações:

$$fa = 13,5Mhz \tag{4.43}$$

$$fsc = \frac{909}{4} f_H \tag{4.44}$$

$$\frac{fa}{6} = 143 f_H \tag{4.45}$$

substituindo a Eq. 4.44 na Eq.4.45, com fsc=3,57561149Mhz, temos:

$$\bullet fa = 13,5Mhz$$

substituindo a Eq.4.45 na Eq.4.44, temos:

$$fsc = \frac{909}{6 \times 143 \times 4} fa \tag{4.46}$$

com fa=13,5 Mhz:

$$\bullet fsc = 3575611,88Hz$$

considerando a faixa de $\pm$10Hz do erro, concluimos que as expressões 4.43, 4.44 e 4.45 são válidas para o sistema PAL-M para a freqüência de 13,5Mhz.

Da Eq.4.45:

$$T_H = 6 \times 143 \times Ta \tag{4.47}$$

$$T_H = 858Ta \tag{4.48}$$

Como o número de amostras totais na linha, 858, é o mesmo do sistema em cores NTSC e sabendo que os tempos de Apagamento Horizontal são iguais, chegamos as mesmas conclusões da Tab.4.7.

Dada a importância de sinais digitalizados na freqüência de 13.5Mhz, pretende-se futuramente fazer os programas de conversão de 8/3fsc e 4fsc para 13.5Mhz e de 13.5Mhz para 8/3fsc e 4fsc, respectivamente.

4.8 Conversão da freqüência de 10Mhz para $4fsc$

As imagens digitalizadas em uma freqüência de 10Mhz correspondem a um quadro de 512X512 amostras.

Considerando que o quadro de amostras ativas para uma freqüência de amostragem de 8/3fsc é de 504X512 amostras, dada a proximidade entre as freqüências, concluimos que: desprezando quatro amostras nas extremidades da imagem de 10Mhz (oito por linha) ao longo das 512 linhas, estaremos aproximando o sinal para uma freqüência de amostragem de 8/3fsc.

Fazendo a alteração de desprezo no programa de conversão de 8/3fsc para 4fsc, geramos o programa de conversão de 10Mhz para 4fsc (**dezqtr.c**), veja apêndice C.

Os resultados da conversão são praticamente os mesmos da conversão de 8/3fsc para 4fsc.

4.9 Conclusões

Os estudos elaborados permitiram a implementação de funções para a conversão de freqüências usando o teorema da amostragem. Para o cálculo de uma amostra (pixel), foram necessárias no máximo 8 amostras do sinal original. A diminuição de amostras para o cálculo foi desejada uma vez que contribui para a redução do tempo de processamento.

A necessidade das funções de conversão são devido a viabilidade para a redução da taxa de bits inerente para sistemas digitalizados.

O resultado obtido, proporciona a inclusão das funções para o Sistema Digital de Visualização de Imagens, contribuindo dessa forma para o desenvolvimento do sistema.

Figura 4.4: SMPTE15 Imagem original a uma freqüência de 4fsc.

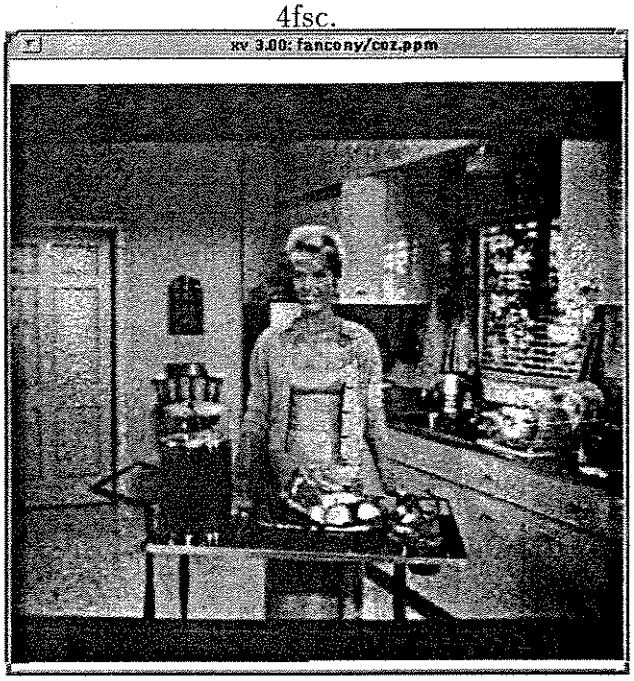

Figura 4.5: SMPTE15 Imagem original a uma freqüência de 4fsc.

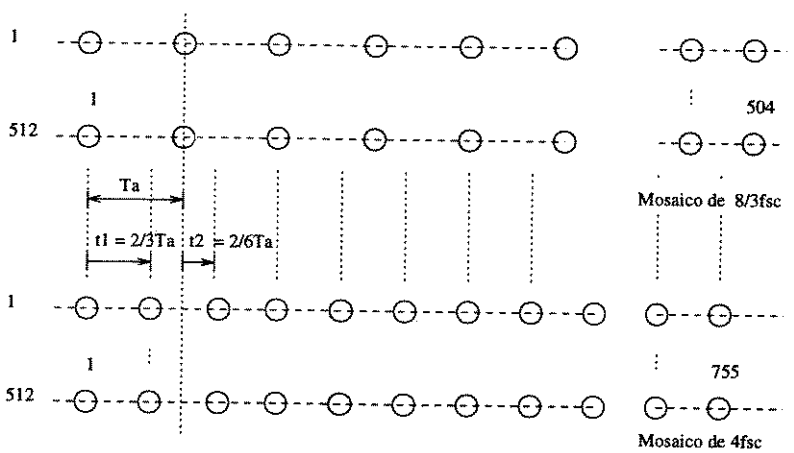

Figura 4.6: Comparação entre os mosaicos de 8/3fsc e 4fsc.

Figura 4.7: SMPTE02 Imagem original a uma freqüência de 8/3fsc.

Figura 4.8: SMPTE02 Imagem processada para a freqüência de 4fsc.

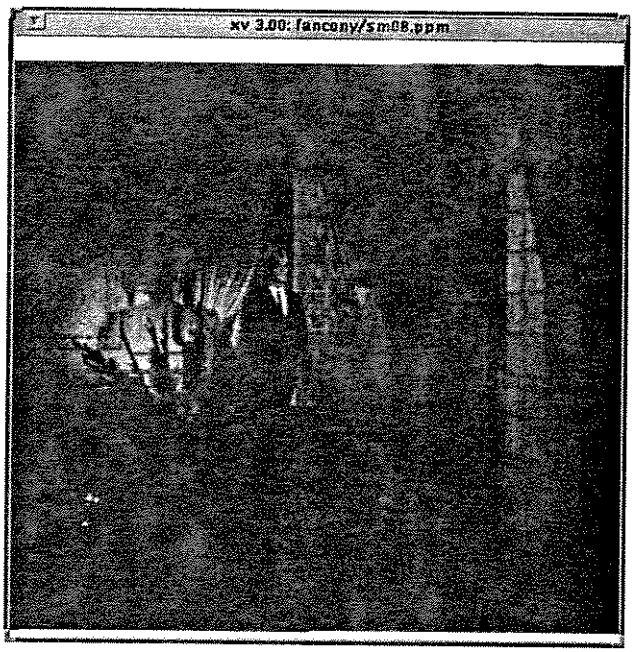

Figura 4.9: SMPTE08 Imagem original a uma freqüência de 8/3fsc.

Figura 4.10: SMPTE08 Imagem processada para a freqüência de 4fsc.

Capítulo 5

Conclusões

Este trabalho aborda dois temas de pesquisa: o Ambiente de Trabalho SDVI e funções de conversão de freqüências.

O primeiro tema é sobre a implementação de uma interface de trabalho para um sistema de Digitalização, Processamento e Visualização de sinais de vídeo, que viemos a denominar como SDVI.

A interface foi implementada através da ferramenta Devguide (OpenWindows Developer's Guide versão 1.1). Apresentamos assim inicialmente um estudo resumido sobre as características da ferramenta com sugestões sobre a literatura correspondente. Tecemos considerações sobre a utilidade da ferramenta, dando início assim pela primeira vez a uma abordagem de grande interesse dentro da área de processamento de sinais. É a primeira abordagem do gênero, por isso o interesse em pormenorizar a ferramenta.

Estudamos conveniêntemente uma série de opções que deveriam constar da interface de forma que satisfizessem todas as atribuições do sistema em implementação iterativa. Ou seja, opções que satisfaçam trabalhos que venham a ser implementados futuramente.

Dado o porte a que se quer chegar com o sistema, justificam-se as inúmeras janelas de notícias que se sobressaiem como "*Função não implementada. Por anexar!* ", informando sobre a não existência da função na opção da janela e que esta, seria anexada a interface quando implementada ou, substituída quando melhorada. Usamos a função *"System"* do compilador C para a execução das operações. Apesar de ser bastante iterativo, uma das grandes conveniências do ambiente de trabalho gerado é o auxílio "ajuda" que fornece para todas as suas janelas através da opção do teclado "Help", bastando portanto, posicionarmos o "mouse" sobre a região da interface em dúvida e pressionarmos a tecla "Help". Abre-se assim uma janela com um texto de auxílio.

Apresentamos por questões técnicas e de auxílio no apêndice em anexo o programa fonte. Nele podemos observar a conexão feita entre janelas de uma mesma interface. A versão do Devguide usada não conecta as janelas entre sí, ficando isso por parte do programador.

Concluimos que para uma primeira fase (as fases serão contínuas, o ambiente crescerá, será realimentado, terá novas versões), o ambiente criado está a altura de proporcionar o desenvolvimento de trabalhos de pesquisa em grupo, tornando-se um polo de aglutinação do grupo

de estudos, quer seja como fonte de informações, processamento, visualização ou digitalização[1] como, destino de algum trabalho ou uma nova técnica realizada por elementos do grupo.

A pesquisa em grupo é uma estratégia que nos leva a muitos e bons resultados; já se constatou em estudos sobre dinâmica de grupo a eficiência de trabalhos assim desenvolvidos. A alternativa para consultas, troca de idéias, experiências, etc. faz com que este projeto inicial seja fomento de novos trabalhos, maior aproveitamento e entrosamento dentro do grupo. Pretendemos alcançar com este tema, o início de uma nova geração de trabalhos dentro do grupo.

Um dos trabalhos fundamentais a ser realizado pelo grupo será a conexão e interfaceamento (software) entre o digitalizador de vídeo e a estação de trabalho (*WorkStation*) no caso com o ambiente SDVI. Então deve-se concluir nessa altura, uma das fases mais importantes do projeto.

O segundo tema aborda a implementação de funções em linguagem C para a conversão de freqüências usando-se para isso a definição do teorema de amostragem.

Tecemos inicialmente considerações sobre alguns detalhes da teoria de amostragem com o intuito de elucidarmos as expressões usadas dentro dos algorítimos desenvolvidos. Desenvolvemos assim os programas de conversão de freqüência de 4fsc (fsc: freqüência de sub-portadora de cor, quadro de 755X512 amostras ativas por linhas) para 8/3fsc (quadro de 504X512 amostras ativas por linhas) programa **qtroito.c**, de 8/3fsc para 4fsc (programa **oitoqtr.c**) e de 10Mhz (quadro de 512X512 amostras ativas por linhas) para 4fsc (programa **dezqtr.c**). Utilizou-se para esse último programa uma aproximação entre o sinal de 10Mhz e o sinal de 8/3fsc; isso se deve pela aproximação existente entre as duas freqüências possibilitando dessa forma usar os recursos do programa oitoqtr.c, garantindo a disposição vertical das amostras no mosaico de freqüências. A aproximação foi feita desprezando as primeiras e as últimas quatro amostras (bordas da imagem).

Foram usadas as freqüências de 4fsc e de 8/3fsc devido ao alinhamento vertical existente ao longo do mosaico que, garante melhor agilidade na implementação de trabalhos relacionados com processamento de sinais.

Para chegarmos aos resultados obtidos foram feitas várias simulações para a definição estatística de vários parâmetros. É que dependendo do tipo de sinal, o tratamento é específico; usamos três tipos de sinais para os testes: um sinal constante, uma rampa e uma senóide. Assim sendo as simulações foram feitas de forma que se otimizassem os resultados achados para esses três sinais, generalizando por conseguinte os resultados.

Para a avaliação objetiva do sinal, enfatizamos a relação sinal/ruído, principal características sobre a semelhança entre sinais. Para isso, através de programa auxiliar, foram gerados sinais de teste (CCIR-I, CCIR-II, OS2, OTM100 e OTM75) e comparados com os sinais processados através da conversão de freqüências. O resultado apresentado é bom e é exibido através de tabelas anexas. É de considerar que os sinais gerados nas freqüências de 4fsc e 8/3fsc não passam por filtros adequados de 4.2Mhz, o que pode resultar em possíveis problemas. Outra característica percebida é que a relação sinal/ruído aumenta de acordo com a menor variação existente entre amostras adjacentes, ou seja, variações não abruptas ao contrário

[1]Interface de entrada/digitalizador do sinal de vídeo em implementação, trabalho em paralelo do grupo de PDI/DECOM.

do que acontece com contornos.

Para a simulação destes sinais foi necessário a implementação de vários programas auxiliares.

Para a avaliação subjetiva foram usadas as imagens do SMPTE ("Society of Motion Picture and Television Engineers") SALA (SMPTE02), SALA -ESCURA (SMPTE08) e COZINHA (SMPTE15), pois essas imagens apresentam muitas variações de detalhes de fundo, variações de luz, variações de ambiente, etc. A classificação foi dada por seis observadores conforme sugere o CCIR, com notas entre 1 a 5 e o resultado é apresentado em tabelas e figuras ao longo do trabalho.

Este trabalho contribui para:

- A formação de um Sistema Digital de Processamento e Visualização de Imagens;
- Aumento da interação entre os elementos do grupo de PDI, proporcionando maior agilidade para o desenvolvimento e fomento de pesquisas;
- Contribui como fonte de informações e de ferramentas de processamento de sinais;
- A redução do número de amostras no processo de conversão de freqüências, tendo como consequência a redução do tempo de processamento para a obtenção do resultado final.

Concluimos assim este trabalho, iniciando por conseguinte a nova fase em que o sistema será explorado por terceiros. Esta exploração tem como intuíto o enriquecimento da interface através de opiniões, de inclusão de programas fontes de trabalhos na área de processamento de sinais de vídeo/imagens, de consultas bem como uma aproximação inerente dentro do grupo de trabalho.

Apêndice para o Devguide OpenWindows Developer's Guide 1.1

Como colocado no Capítulo 2, O Devguide é usado para fazer a montagem dos elementos pertencentes a uma interface, através do manuseio de representações visuais destes da janela principal, Fig.2.1, para a "*workspace*" da estação. Na interface em construção, podem-se associar janelas, áreas de controle, botões, menus e outros elementos pertencentes a OPEN LOOK UI.

Composição da janela Devguide

A janela Devguide contém controles e informações necessárias para operar o Devguide e os blocos dos elementos necessários a construção das interfaces. Os controles estão localizados no topo, os blocos no meio e as informações acerca dos elementos na base da janela respectivamente.

I- Os controles da janela Devguide são:

- **File** contém um menu com os ítens Load, Save, Save As... e Close.

  Load - Carrega um arquivo previamente salvo no formato GIL (Graphical Interface Language file). Para carregar esses arquivos pode-se usar também o gerenciador de arquivos, para isso, sobrepõe-se o ícone do arquivo de extensão ".G" (GIL) proveniente do gerenciador de arquivos, sobre a janela do Devguide. Rapidamente abre-se o arquivo para operação. Caso exista um outro arquivo aberto no momento da sobreposição do ícone, inicialmente este se fechará sendo que, se por ventura, não estiver salvo o Devguide questionará sobre o salvar ou não do arquivo.

  Save - Grava modificações feitas em uma interface anteriormente salva. Para salvar pela primeira vez a interface usa-se a opção Save As.... Quando usada, o Devguide salva o arquivo com a extensão ".G" e gera o *backup* do arquivo anterior com a extensão ".BAK". Por exemplo existindo um arquivo "face.G" para o qual foram feitas modificações, no uso da opção Save, salvam-se essas modificações para o arquivo "face.G" e gera-se o arquivo *backup* "face.G.BAK" como sendo o anterior sem as modificações recentes.

  Save As... - Salva uma nova interface ou permite salvar uma interface já existente com um outro nome. Ao salvar o arquivo com um determinado nome o Devguide acrescenta a extensão ".G" a este. Se por acaso tentar-se salvar a interface com o nome de um arquivo já existente, o Devguide questionará por meio de uma janela *(notice)* sobre a sobreposição

dos arquivos, caso se der continuidade ao processo, gera o arquivo *backup* do anterior e salva o arquivo atual com o nome fornecido. O arquivo salvo tem o formato GIL, um arquivo texto que descreve os parâmetros de cada elemento na interface do usuário e a conexão entre os elementos. Mais informações sobre o arquivo GIL e seu protocolo podem ser encontradas no Apêndice A da referência [1].

Close - Fecha a interface atualmente aberta no Devguide, removendo todos os elementos da área de trabalho. Caso não se tenha salvo a interface, aparecerá uma mensagem alertando para o fato, seguida das opções: "Discard Changes" que ignora as mudanças feitas e "Cancel" que cancela o fechamento da interface possibilitando que se salvem todas as mudanças.

• **View** Contém um menu com o ítem - Dismissed Windows.

Dismissed Window - Permite recolocar na área de trabalho "workspace", janelas anteriormente fechadas. Durante a montagem da interface, uma janela criada pode ser momentaneamente desprezada e fechada através da opção *dismiss* do seu menu.

• **Edit** Contém um menu com os ítens: Cut, Copy, Paste, Delete e Undo.

Cut - Remove da área de trabalho um elemento da interface selecionado e o coloca em uma memória *"clipboard"*. Toda vez que esta opção for usada, substitui o elemento anterior do *"clipboard"* pelo atualmente selecionado.

Copy - Copia qualquer elemento selecionado e o coloca na memória *"clipboard"*. Toda vez que esta opção for usada substitui o existente no *"clipboard"*, pelo elemento selecionado na operação.

Paste - Cola o elemento atualmente existente na memória *"clipboard"* para a área indicada pelo mouse na interface corrente. Caso exista uma janela gravada no *"clipboard"* esta será aberta na *"workspace"*. Se existir uma área de controle ou um painel no *"clipboard"* deve-se primeiro selecionar uma janela antes de se usar a opção. Caso existirem vários elementos de interface no *"clipboard"*, deve-se primeiro selecionar a área de controle para onde serão gravados esses elementos. Existe uma hierarquia a seguir para implementação da interface do usuário.

Delete - Apaga todo elemento selecionado sem colocá-lo em memória. Com o uso desta operação não se terá retorno do elemento através da opção Paste.

Undo - Desfaz a última operação feita. Usa-se para recuperar elementos apagados através de Delete e Cut ou para remover elementos colados anteriormente a operação.

• **Properties** Contém um menu com ítens que abrem janelas para edição de propriedades de elementos individuais dentro da interface do usuário. Veja Fig.2.2.

Selection - É a opção *default*, abre a janela de propriedades do elemento selecionado. O grupo de opções existentes no meio da janela são ítens de elementos. Cada item abre uma janela de propriedades para o tipo de elemento da interface do usuário (User Interface Element- UIE) escolhida. Veja exemplo para a edição da *Base Window* na Fig.2.3. Os tipos de elementos são: *Base Windows, Pop-up Windows, Control Areas, Canvas Panes,*

Term Panes, Text Panes, Buttons, Messages, Settings, Text Fields, Sliders, Gauges e Lists.

Help - Abre a janela de edição para textos de auxílio.

Menus - Abre a janela de Menus, onde se criam e editam menus.

• **Build/Test** Opções para colocar o Devguide no modo montagem ou teste. Com a opção *"default" Built*, os elementos são associados de forma a criar-se a Interface do Usuário (UI). Uma vez criada, a opção *Test* permite simular a interface manipulando sua estrutura funcional, sem que haja qualquer tipo de interação com os programas fontes a adicionar para o ambiente.

II- Informação acerca dos elementos:

A informação acerca dos elementos é mostrada na base da janela e permite:

Element - Campo disponível para o nome do elemento indicado pelo ponteiro do mouse.

Position - Campo disponível para localizar a posição do elemento indicado pelo ponteiro do mouse. Se o elemento for uma janela a referência da localização será a extremidade superior esquerda da *workspace* medida em *pixels*. Caso seja um elemento interno a janela, então esta será usada como referência.

Size - Campo destinado ao tamanho em *pixels* do elemento apontado pelo mouse.

Pointer - Mostra as coordenadas X (horizontal) e Y (vertical) do ponteiro do mouse dentro de uma janela de interface do usuário (UIWindow). As coordenadas são medidas em *pixels* a partir da extremidade superior esquerda da Janela de Interface do Usuário (UIW) ou painel.

III- Os elementos da Interface do usuário: No centro da janela Devguide, encontram-se representados em miniatura os blocos dos elementos que podem ser adicionados à interface do usuário. Existem vários elementos, como podemos observar no centro da Fig.2.1, temos:

Base Windows

No OpenWindows, essa janela, define uma seção da área de trabalho, onde um aplicativo fornece informações, executa funções e oferece controles para o usuário. As interfaces devem ser inicializadas através de uma *Base Window*.

Janela de propriedades da Base Window

Para editar as propriedades de uma *Base Window*, o usuário deve selecionar primeiro a *Base Window* (UIE), como anteriormente mencionado, e abrir a sua janela de propriedades. Dentro da janela de propriedades existem vários campos onde o usuário define a janela criada. Os campos para a *Base Window* são: *Base Windows:* - Lista de elementos do mesmo tipo no topo da janela. *Name:* - Nome da janela para o código em C a ser gerado. *Label:* - Rótulo para o topo da janela. *Retangle: W: H:* - Largura e altura em pixels da janela. *Initial State: Open* e *Iconic* - Estado inicial para a janela, se aberta ou em ícone. *Footer: Present* e *Not Presente* - Borda presente ou não na base da

janela. A borda é usada para mensagens de controle. *Size: Adjustable* e *Fixed* - Permite ser ajustável ou fixo o tamanho da janela. *Icon:* - Campo para o nome do arquivo de imagem que forma o ícone. *Icon Type: Normal* e *Borderless* -Tipo de ícone. Se normal ou com borda. *Icon Mask:* - Campo para o arquivo máscara do ícone. *Event Handler:, Keyboard Events, Mouse Events* - Campo para nome de eventos a se realizar e os seus tipos, se do teclado ou do mouse respectivamente. *Foreground... e Background...* - Dois campos para escolha das cores de fundo e de frente da janela. *Apply* e *Reset* - Botões para aplicar e reiniciar as propriedades da janela.

Proceguindo:

Pop-up Windows

No OpenWidows uma *Pop-up Window*, como a *Base Window*, define uma seção da área de trabalho onde um aplicativo fornece informações, realiza funções e, oferece controles para o usuário. A janela *Pop-up* é uma auxiliar para a *Base Window* e nunca aparece como principal em um programa. Ou seja, quando se utiliza um programa, primeiro se abre uma *Base Window* que geralmente permanece aberta até o término do programa ou quando se fecha para o seu *Icone* correspondente. Janelas *Pop-up*, típicamente aparecem quando são necessárias mais informações ou controles. Desaparecem logo de imediato. A janela de propriedades oferece os seguintes campos (funções já mencionadas anteriormente): *Pop-up Windows:* - Lista com elementos da mesma espécie.*Name, Label, Rectangle: W: H:, Initial State: Open* e *Iconic, Footer: Present* e *Not Presente, Size: Adjustable* e *Fixed, Pushpin: Out* e *In* - Estado do alfinete da janela, se dentro ou fora. *Done Handler:* - Campo com o nome da rotina chamada quando é desfeita a janela. *Event Handler:, Keyboard Events, Mouse Events, Foreground..., Background...* e finalmente os botões: *Apply* e *Reset* para aplicar e reiniciar as propriedades da janela.

Control Areas

Uma área de controle no OpenWidows, *Control Area*, é a região de uma janela onde o programa exibe controles como os: "*Buttons, Settings Sliders ...*". A área de controle deve ser colocada dentro de uma janela e sobre ela os controles. A janela de propriedades tem os seguintes campos: *Control Areas:, Name, Rectangle: x: y: W: H:* - Acrescenta-se neste caso as coordenadas em pixels x e y que tenhem como referência a extremidade superior esquerda da área de controle. *Show Border: No* e *Yes* - Mostra ou não a borda da área de controle. *Event Handler:, Keyboard Events, Mouse Events, Foreground..., Background...* e finalmente os botões: *Apply, Reset* e *Help Text...* . O *Help Text* abre o editor de textos de ajuda para a área de controle.

Canvas Panes

No OpenWidows, *Canvas Pane* é a região de uma janela onde o programa apresenta gráficos. O usuário pode exibir nessa região seus próprios gráficos ou editar os já existentes. Essa região, geralmente possui cursores tanto na vertical como na horizontal, permetindo que o usuário se guie através destes para a visualização de uma imagem cujo tamanho é superior ao da região. A janela de propriedades possui as seguintes áreas: *Canvas Panes:, Name:, Rectangle: x: y: W: H:, Horizontal Scrollbar: Not Present* e

Present, Vertical Scrollbar: Not Present e *Present* -Presença ou não de cursores na horizontal e vertical respectivamente. *Menu Name:* - Área para o nome de algum menu atribuído a essa janela. *Repaint handler:* - Nome da rotina que o XView chama quando a imagem é danificada. *Drawing Model: XView X Windows* e *PostScript* - Modos gráficos possíveis para a janela. *Event Handler:, Keyboard Events, Mouse Events, Foreground..., Background...* e finalmente os botões: *Apply, Reset* e *Help Text...* . O *Help Text* abre o editor de textos de ajuda para a janela *Canvas Panes.*

Term Panes

No OpenWidows, *Term Panes* é a região de uma janela onde o programa apresenta o *UNIX Shell* para o usuário. Apresenta o *prompt* onde o usuário escreve comandos UNIX e trabalha diretamente com o sistema operacional *SunOS.* A janela de propriedades desta região tem os seguintes campos:*Term Panes:, Name:, Rectangle: x: y: W: H:, Show Border: Yes* e *No, Event Handler:, Keyboard Events, Mouse Events, Foreground..., Background...* e os botões: *Apply, Reset* e *Help Text...* .

Text Panes

No OpenWindows a *Text Pane* é uma região da janela onde o programa exibe texto e onde, por conseguinte, o usuário pode editar novos textos. Para que o usuário possa correr ao longo do texto contido na *Text Pane*, existe um cursor posicionado verticalmente no lado direito da tela. A janela de propriedades tem as seguintes áreas: *Text Panes:* - Na parte superior. *Name:, Rectangle: x: y: W: H:, Show Border: Yes* e *No, Operation: Read-write* e *Read-only* - Tipo de operação. Escrita e leitura ou se somente leitura. *Event Handler:, Keyboard Events, Mouse Events, Foreground..., Background...* e os botões: *Apply, Reset* e *Help Text...* .

Buttons

No OpenWindows, o *Button* é um botão de controle dentro da área de controle, que inicializa uma ação ou apresenta um menu se pressionado. Quando se é amarrado um menu para o botão ele é denominado *menu button.* A janela de propriedades apresenta os seguintes campos: *Buttons:* - No topo. *Name:, Label Is: Text* ou *Glyph Filename* - Se existe um rótulo em forma de texto ou em imagem. *Location: x:* e *y:* - Ponto de localização tendo como referência a extremidade superior esquerda da área de controle onde se encontra o botão. *Type: Full Size* e *Abbreviated Menu* - O primeiro é o botão padrão em forma de uma caixa com um rótulo dentro. O segundo tem a forma abreviada, sendo uma pequena seta para baixo com o nome do botão nas proximidades. *Menu Name, Notify Handler, Event Handler:, Keyboard Events:, Mouse Events:, Color...* e os botões: *Apply, Reset* e *Help Text...* .

Messages

No OpenWindows, *Messages* é um controle que exibe mensagens para o usuário em texto no modo de leitura ou como uma imagem. Pode se usar *message* para adicionar textos ou imagens na área de controle. A janela de propriedades apresenta os seguintes campos: *Messages:, Name:, Label:, Label Is: Text* ou *Glyph Filename* - Se existe um rótulo em forma de texto ou em imagem. *Label Font: Bold* ou *Normal* - Tipo de fonte para o texto

da messagem, se enfatizada ou normal. *Location: x:* e *y:* - Ponto de localização tendo como referência a extremidade superior esquerda da área de controle onde se encontra a janela. *Event Handler:, Keyboard Events:, Mouse Events:, Color...* e os botões: *Apply, Reset* e *Help Text...* .

Settings

O *Setting* é um controle que oferece diferenciadas alternativas de escolha de funções. Existem diferentes tipos de *settings: Exclusive, nonexclusive, check boxes e stacks.* Este elemento deve ser colocado dentro de uma área de controle. A janela de propriedade tem os seguintes campos: *Settings:, Name:, Label:, Label Is: Text* ou *Glyph Filename, Location: x:* e *y:, Type:* - Opção para escolha de um dos quatro tipos de *settings. Rows/Columns:* - Posição dos *settings*, se em filas ou colunas respectivamente. *Notify Handler:, Event Handler:, Keyboard Events:, Mouse Events:, Color..., Choices:* - Janela de inserção e edição de opções de controle. *Label:, Label Is:, color...* - Rótulo, forma de rótulo e cor do controle inserido respectivamente, e os botões: *Apply, Reset* e *Help Text...* .

Text Field

Text Field é um controle no OpenWindows, que solicita do usuário um conjunto de palavras na forma de texto ou apresenta uma série de palavras para que o usuário edite em determinado campo. São tipicamente usados para pedir valores numéricos ou arquivos acompanhados de seus respectivos diretórios. Somente pode ser colocado dentro de áreas de controle. A janela de propriedades apresenta os seguintes campos: *Text Fields:, Name:, Label:, Label Is: Text* ou *Glyph Filename, Location: x:* e *y:, Field Type: Alphanumeric* ou *Numeric* - O campo de texto apresentado pode ser alfanumérico, para caracteres, ou numérico, para valores numéricos. *Range: Min: Max:* - Limites de faixa inferior e superior quando usada a opção numérica para o campo anterior. *Field Length:, Stored Length:* - Tamanho em caracteres do campo de texto selecionado e tamanho em caracteres da capacidade de armazenamento do campo de interface de texto de usuário selecionada. *Operation: Read-write* e *Read-only, Notify Handler:, Event Handler:, Keyboard Events:, Mouse Events:* e os botões *Color..., Apply, Reset* e *Help Text...* .

Sliders

No OpenWindows, *Sliders* é um controle dentro da área de controle que posiciona um cursor em determinado valor dentro de uma faixa de valores. Para a escolha de um valor, o usuário corre com o cursor, usando para isso o mouse, ao longo da faixa de dados existente. A janela de propriedades de um *slider* tem os seguintes campos: *Sliders:* - No topo, *Name:, Label:, Label Is: Text* ou *Glyph Filename, Location: x:* e *y:, Orientation: Horizontal* ou *Vertical* - Posição em que se movimento o cursor. *Value: Present* ou *Not Present* - Presença ou ausência do valor corrente do cursor. *Range: Show* ou *Hide, Min:* e *Max:* - Presença ou ausência dos limites de faixa nas extremidades do cursor, com os valores mínimo e máximo respectivamente. *End Boxes: Present* e *Not Present* - Presença ou ausência de marcas nas extremidades do cursor. *Ticks:* - Permite a presença ou ausência de valores ao longo do cursor. *Notify Handler:, Event Handler:, Keyboard Events:, Mouse Events:* e os botões *Color..., Apply, Reset* e *Help Text...* .

Gauges

Ao contrário de *Sliders*, o *Gauge* fornece para o usuário, determinado valor dentro de uma faixa de valores possíveis. É simplesmente um processo de leitura. Esse elemento é usado dentro de uma área de controle. A janela de propriedades apresenta os seguintes campos: *Gauges:, Name:, Label:, Label Is: Text* ou *Glyph Filename, Location: x:* e *y:, Orientation: Horizontal* ou *Vertical, Range: Show* ou *Hide, Min:* e *Max:* - Presença ou ausência dos limites de faixa nas extremidades do cursor, com os valores mínimo e máximo respectivamente. *Ticks:* - Permite a presença ou ausência de valores ao longo do cursor. *Event Handler:, Keyboard Events:, Mouse Events:* e os botões *Color..., Apply, Reset* e *Help Text...* .

Scrolling Lists

Apresenta para o usuário uma lista em forma de texto na posição vertical, corrida por um cursor com ítens que o usuário pode escolher usando o ponteiro do mouse. É tipicamente usada para acomodar confortavelmente um conjunto amplo de opções em forma de texto em um espaço limitado. Este elemento é posicionado dentro de uma área de controle. A janela de propriedades tem os seguintes campos: *Lists:, Name:, Label:, Label Is: Text* ou *Glyph Filename, Location: x:* e *y:, Rows:* - Número máximo de linhas de texto presentes na lista.Controla o tamanho da janela. *Orientation: Read-write* ou *Read-only, Choice: Required* ou *Not Required* - A lista exige que pelo menos um ítem seja escolhido ou não necessariamente. *Choices: Exclusive* ou *Nonexclusive* - A lista permite a escolha de um único ítem ou a escolha de vários, para a primeira e segunda opção respectivamente. *Menu Name: Notify Handler:, Event Handler:, Keyboard Events:, Mouse Events:* e os botões *Color..., Apply, Reset* e *Help Text...* .

Menus

A janela de propriedades de um menu tem os seguintes campos:*Menus:* com os botões de opção *Creat* - para criar menus e *Edit* - para edição. *Name:, Title:* - Título do menu corrente. *Columns:* - Número de colunas usadas para mostrar os ítens. *Menu Is: Pinnable* ou *Not Pinnable* - Se a janela tem ou não o pino. *Handler:* - Nome para a função de monitoração das opções de menu criadas. *Items: Creat* e *Edit* - Campo de opções de ítens com os botões para criar e editar. Para isso temos ainda os campos: , *Label:, Label Is: Text* ou *Glyph Filename, Item Is: Normal* ou *Default,* o botão *Color...* , O *Submenu:* - Campo para o nome de um sub-menu amarrado a determinado ítem. *Handler:,* E os botões *Apply* e *Reset* .

Apêndice

São listados a seguir os:

1 Programas de conversão de freqüências:

- qtroito.c
- oitoqtr.c
- dezqtr.c

2 Programas em C da interface:

- Makefile
- sdviP.c
- FSProc.c
- sdvipro.c

```
/* Programa qtroito.c                                                    */
/* Le um arquivo na frequencia de 4fsc (755X512 amostras ativas em ASC) e */
/* converte o mesmo para  8/3fsc (503X512 amostras ativas em ASC.        */
/* Usa a media ponderada acrescida aos coeficientes da funcao sample()   */

#include <stdio.h>
#include <math.h>
#define PI 3.1415927
#define Colqto  755          /* 755 Numero de colunas do sinal em 4fsc    */
#define Coloit  503          /* 503 Num.   de colunas do sinal em 8/3fsc  */
#define Linhas  512      /* 512 Num. de linhas dos sinais em 4fsc e 8/3fsc */
#define Ext      3           /* Num. de contribuicoes para o somatorio da */
#define Ext1     1           /* Sample. "n" varia de -Ext a +Ext+1;       */
#define st       1           /* delta, variacao para confirmacao de sinal */
#define zero     0
float qtofsc[Colqto+2*Ext];  /* Amostras ativas reais 4fsc, mais Ext amos */
                             /* tras acrescidas aos extremos para corre-  */
                             /* cao das primeiras amostras de 8/3fsc      */
float oitofsc[Coloit];       /* Amostras ativas reais 8/3fsc, uma linha   */
float temp[Ext + 1];         /* Coeficientes                              */
float temp1[Ext1+1];         /* da funcao sample                          */

main (argc,argv)
   int argc;
   char *argv[];
   {
   void sa();
   int e,n1,j,jj,nzeros,k,Exttemp;
   int soma;                 /* Uso de reciprocidade dos coeficientes da sa */
   int N;                    /* N-esima amostra ativa de 8/3fsc 0<=N<=502   */
   int n;                    /* n-esima amostra ativa de 4fsc 0<=n<=754     */
   int l;                    /* Numero de linhas em 8/3 e 4fsc 1<=l<=512    */
   float R;                  /* Posicao da N-esima amostra de 8/3 em 4fsc   */
   float kk;
   float B;                  /* Fator de correcao conforme Ext              */
   float Extind = 0.0;       /* Fator de correcao dependente de Ext         */
   float s1, s2, sinal;      /* Identificam sinal de incremento/decremento  */
   float s3, s4, s5, s6, s7,sm;   /* Identificam tipo de sinal, se tende    */
   float d3, d4, d5, d6, d7;      /* a senoide, constante ou triangular     */
   FILE *entqto;             /* Arquivo de entrada 4fsc, 755X512 amostras   */
   FILE *saidoito;           /* Arquivo de saida 8/3fsc, 503X512 amostras   */
   int  *cc;                 /* Inteiro correspondente ao sinal 8/3fsc      */
   cc = (int*)malloc(sizeof(int));   /* Espaco de memoria alocada para cc   */
     if(argc!=3){                    /* Se o argumento for diferente de 3,  */
                                     /* imprime a forma de uso na tela      */
     printf("\nModo de usar: qtroito Sinal_4fsc Sinal_8/3fsc");
     printf("\nargumentos: Sinal_4fsc: Arquivo de entrada \n");
     printf("            Sinal_8/3fsc: Arquivo processado \n");
     exit(1);
     }
     if((entqto = fopen(argv[1],"r")) == NULL){
       printf("Erro na abertura do arquivo de entrada:%s\n", argv[1]);
       exit(1);
       }
     if((saidoito = fopen(argv[2],"wa")) == NULL){
       printf("Erro na abertura do arquivo de saida: %s\n",argv[2]);
       exit(1);
       }
     else{
       B = Extind/127;
```

```
        for(k=0;k<=(Colqto+2*Ext-1);k++){ qtofsc[k] = 0.0; }
        for(k=0;k<=(Coloit-1);k++){ oitofsc[k] = 0.0; }
        for(k=0;k<=Ext;k++){ temp[k] = 0.0; }
        for(k=0;k<=Ext1;k++){ temp1[k] = 0.0; }
        sa(Ext,temp);
        sa(Ext1,temp1);
        for(l=1;l<=Linhas;l++){                 /* Para todas as linhas de 4fs */
         for(n=Ext;n<=(Colqto+Ext-1);n++){      /* Le uma linha do sinal 4fs   */
           fscanf(entqto,"%4d",&e);
           qtofsc[n] = e;
           }
         for(k=0;k<=(Ext-1);k++){               /* Preenche as Ext amostras ...*/
           qtofsc[k] = (qtofsc[Ext+1]+qtofsc[Ext+2])/2.0; /*...iniciais e...*/
           }                                    /* ...finais com a media das...*/
         for(k=Colqto+Ext;k<=Colqto+2*Ext-1;k++){  /*...amostras de extre...*/
           qtofsc[k] = (qtofsc[Ext-2]+qtofsc[Ext-1])/2.0; /*...midade da ...*/
           }                                 /*   amostra a calcular em 8/3fsc */
         for(N=0;N<=(Coloit-1);N++){         /* Calculo das amostras de 8/3fsc */
          R = N*1.5 + Ext;                   /* N-esima amostra de 8/3fsc.     */
          j = R;                             /* R, posicao da amostra em 4fsc  */
          if(fmod(R,(float)j)==0){
             oitofsc[N] = qtofsc[j];         /* Amostragem coincide            */
             }
             else{                           /* Calculo atraves da sample()    */

/* Verifica o tipo de sinal p/ incrementar/decrementar o fator de correcao */

               s1 = qtofsc[j] - qtofsc[j-1];
               s2 = qtofsc[j-1] - qtofsc[j+2];
               d3 = s1;
               if(d3>0.0){s3=1.0;} else{s3=-1.0;}
               d4 = qtofsc[j-1] - qtofsc[j-2];
               /* if(d4>0.0){s4=1.0;} else{s4=-1.0;} */
               d5 = qtofsc[j+1] - qtofsc[j]  ;
               if(d5>0.0){s5=1.0;} else{s5=-1.0;}
               d6 = qtofsc[j+2] - qtofsc[j+1];
               if(d6>0.0){s6=1.0;} else{s6=-1.0;}
               d7 = qtofsc[j+3] - qtofsc[j+2];
               /* if(d7>0.0){s7=1.0;} else{s7=-1.0;} */
               nzeros = 0;
               d3 = fabs(d3); if(d3 == zero){nzeros++;}
               d4 = fabs(d4); if(d4 == zero){nzeros++;}
               d5 = fabs(d5); if(d5 == zero){nzeros++;}
               d6 = fabs(d6); if(d6 == zero){nzeros++;}
               d7 = fabs(d7); if(d7 == zero){nzeros++;}
               if(nzeros >= 3){       /* Calculo pela media (d1=d2=d4=d5=0) */
                 oitofsc[N] = (qtofsc[j] + qtofsc[j+1])/2.0;
                 }
                 else{
                   sm = (d3 + d5 + d6)/3.0;
                   d3 = fabs(d3 - sm);
                   d5 = fabs(d5 - sm);
                   d6 = fabs(d6 - sm);
                   if(d3<st && d5<st && d6<st && s3 == s5 && s3 == s6){
                     jj = Ext1;                              /* d3=d5=d6 SP2 */
                     soma = 0;
                     for(n=(j-Ext1);n<=(j+Ext1+1);n++){
                        oitofsc[N] = qtofsc[n]*temp1[jj] + oitofsc[N];
                        if(jj != 0 && soma == 0){ jj--; }
                        else{
                           if(jj==0 && soma == 0){ jj = -1; soma = 1; }
```

```
                        jj++;
                        }
                    }
                }
                else{
                  jj = Ext;
                  soma = 0;
                  for(n=(j-Ext);n<=(j+Ext+1);n++){
                     oitofsc[N] = qtofsc[n]*temp[jj] + oitofsc[N];
                     if(jj != 0 && soma == 0){ jj--; }
                     else{
                       if(jj==0 && soma == 0){ jj = -1; soma = 1; }
                       jj++;
                       }
                     }
                     if(s1*s2>0.0){sinal=1.0;} else{sinal=-1.0;}
                     oitofsc[N]=oitofsc[N]+sinal*B*(qtofsc[j]-qtofsc[j+1]);
                  } /* conforme sinal decrementa ou incrementa fator      */
                    /* de correcao                                        */
                }
            }
            *cc = faprox(oitofsc[N]); /* Transforma real para inteiro     */
            oitofsc[N] = 0;
            fprintf(saidoito,"%4d",*cc);   /* Imprime a saida de 8/3fsc  */
            } /* for N */
        } /* for I */
        fclose(saidoito); fclose(entqto);
       } /* else */
}


/* Funcao sa(), calcula para cada contribuicao do somatorio, o valor da    */
/* sample correspondente                                                   */
void sa(Extt, tempp)
     int Extt;
     float *tempp;
     {
     int n1;
     float kk = 0.0;
     float spc = 0.0;          /* Acumulado da funcao sample()             */
     float Er  = 0.0;          /* Erro do valor acumulado da funcao sample */
     for(n1=1;n1<=Extt+1;n1++){
        kk = (PI/2.0-n1*PI);
        tempp[n1-1] = (sin(kk))/kk;
        spc       = tempp[n1-1] + spc;
        }
        Er = 0.5 - spc;          /* Efetua a correcao das contribuicoes  ...*/
        for(n1=1;n1<=Extt+1;n1++){ /* da sample atraves da media ponderada */
            tempp[n1-1] = tempp[n1-1]*Er/spc + tempp[n1-1];
            }
     }


/* Funcao faprox(), aproximacao para inteiro mais proximo                  */
faprox(r)
   float r;
   {
   float ri, rf;
   ri = 0.5 + r;
   rf = floor((float)ri);
        return(rf);
```

```
}
```

```
/* Programa oitoqtr.c                                                    */
/* Le um arquivo na frequencia de 8/3fsc (504X512 amostras ativas em ASC)  */
/* e converte o mesmo para  4fsc (755X512 amostras ativas em ASC).         */
/* Usa a media ponderada acrescida aos coeficientes da funcao sample()     */

#include <stdio.h>
#include <math.h>
#define PI 3.1415927
#define Colqto  755           /* 755 Numero de colunas do sinal em 4fsc    */
#define Coloit  503           /* 503 Num.   de colunas do sinal em 8/3fsc  */
#define Linhas  512       /* 512 Num. de linhas dos sinais em 4fsc e 8/3fsc */
#define Ext      2            /* Num. de contribuicoes para o somatorio da */
#define Ext1     1            /* Sample. "n" varia de -Ext a +Ext+1;       */
#define st       1            /* delta, variacao para confirmacao de sinal */
#define zero     0
float oitofsc[Coloit+2*Ext+1];  /* Amostras ativas reais 8/3fsc, mais Ext    */
                           /* amostras acrescidas aos extremos para cor */
                           /* recao das primeiras amostras de 4fsc      */
float qtofsc[Colqto];      /* Amostras ativas reais em 4fsc, uma linha  */
float temp1[2*Ext1 + 2];   /* Coeficientes da funcao Sample()           */
float temp2[2*Ext+2];
float temp3[2*Ext1+2];
float temp4[2*Ext+2];
void sa();
int faprox();

main (argc,argv)
   int argc;
   char *argv[];
   {
   int e,n1,j,jj,nzeros,k,Exttemp,difint,sinal2;
   float N;                /* Posicao da R-esima amostra de 4fsc em 8/3fsc*/
   int n;                  /* n-esima amostra ativa de 8/3fsc 0<=n<=503   */
   int l;                  /* Numero de linhas em 8/3 e 4fsc 1<=l<=512    */
   int R;                  /* R-esima amostra ativa de 4fsc, 0<=R<=754    */
   float kk;
   float B;                /* Fator de correcao conforme Ext              */
   float Extind = 0.0;     /* Fator de correcao dependente de Ext         */
   float s1, s2, sinal;    /* Identificam sinal de incremento/decremento  */
   float s3, s4, s5, s6, s7,sm;   /* Identificam tipo de sinal, se tende   */
   float d3, d4, d5, d6, d7;      /* a senoide, constante ou triangular    */
   FILE *entoito;          /* Arquivo de entrada 8/3fsc, 504X512 amostras */
   FILE *saidqto;          /* Arquivo de saida 4fsc, 755X512 amostras     */
   int  *cc;               /* Inteiro correspondente ao sinal 4fsc        */
   cc = (int*)malloc(sizeof(int));   /* Espaco de memoria alocada para cc  */
     if(argc!=3){                    /* Se o argumento for diferente de 3, */
                                     /* imprime a forma de uso na tela     */
     printf("\nUso: oitoqtrt Orig_8/3fsc Pro_4fsc ");
     exit(1);
     }
     if((entoito = fopen(argv[1],"r")) == NULL){
       printf("Erro na abertura do arquivo de entrada:%s\n", argv[1]);
       exit(1);
       }
     if((saidqto = fopen(argv[2],"wa")) == NULL){
       printf("Erro na abertura do arquivo de saida: %s\n",argv[2]);
       exit(1);
       }
     else{
       B = Extind/127;
```

```
        for(k=0;k<=(Coloit+2*Ext);k++){ oitofsc[k] = 0.0; }
        for(k=0;k<=(Colqto-1);k++){ qtofsc[k] = 0.0;}
        for(k=0;k<=2*Ext+1;k++){ temp2[k] = 0.0; temp4[k] = 0.0;}
        for(k=0;k<=2*Ext1+1;k++){ temp1[k] = 0.0; temp3[k] = 0.0; }
        sa(Ext,temp2,1);
        sa(Ext,temp4,2);
        sa(Ext1,temp1,1);
        sa(Ext1,temp3,2);
        for(l=1;l<=Linhas;l++){              /* Para todas as linhas de 8/3fs */
         sinal2 = 1;
         for(n=0;n<=(Coloit);n++){           /* Le uma linha do sinal 8/3fsc  */
           fscanf(entoito,"%4d",&e);
           }
         for(k=0;k<=(Ext-1);k++){              /* Preenche as Ext amostras com */
           oitofsc[k] = oitofsc[Ext+1];        /* a segunda amostra de 8/3fsc  */
           }
         for(k=Coloit+Ext;k<=Coloit+2*Ext-1;k++){ /* Preenche as Ext amostras*/
           oitofsc[k] = oitofsc[Ext-2];    /* finais com a penultima de 8/3fsc*/
           }
         for(R=0;R<=(Colqto-1);R++){       /* Calculo das amostras de 4fsc   */
          N = (float)R/1.5 + Ext;                  /* N-esima amostra de 8/3fsc.
  */
          j = N;                           /* R, posicao da amostra em 4fsc  */
          if(fmod(N,(float)j)==0){
             qtofsc[R] = oitofsc[j];       /* Amostragem coincide            */
             }
             else{                         /* Calculo atraves da sample()    */

/* Verifica o tipo de sinal p/ incrementar/decrementar o fator de correcao */

               s1 = oitofsc[j] - oitofsc[j-1];
               s2 = oitofsc[j-1] - oitofsc[j+2];
               d3 = s1;
               if(d3>0.0){s3=1.0;} else{s3=-1.0;}
               d4 = oitofsc[j-1] - oitofsc[j-2];
               /* if(d4>0.0){s4=1.0;} else{s4=-1.0;} */
               d5 = oitofsc[j+1] - oitofsc[j]  ;
               if(d5>0.0){s5=1.0;} else{s5=-1.0;}
               d6 = oitofsc[j+2] - oitofsc[j+1];
               if(d6>0.0){s6=1.0;} else{s6=-1.0;}
               d7 = oitofsc[j+3] - oitofsc[j+2];
               /* if(d7>0.0){s7=1.0;} else{s7=-1.0;} */
               nzeros = 0;
               d3 = fabs(d3); if(d3 == zero){nzeros++;}
               d4 = fabs(d4); if(d4 == zero){nzeros++;}
               d5 = fabs(d5); if(d5 == zero){nzeros++;}
               d6 = fabs(d6); if(d6 == zero){nzeros++;}
               d7 = fabs(d7); if(d7 == zero){nzeros++;}
               if(nzeros >= 3){  /* d1=d2=d4=d5=0, Bordas, regioes uniformes */
                 if(sinal2==1){
                   qtofsc[R] = (2.0/6.0)*oitofsc[j] + (2.0/3.0)*oitofsc[j+1];
                   }
                   else{
                    qtofsc[R] = (2.0/3.0)*oitofsc[j] + (2.0/6.0)*oitofsc[j+1];
                    }
                 sinal2 = (sinal2==1)?2:1;
                 }
                 else{
                   sm = (d3 + d5 + d6)/3.0;
                   d3 = fabs(d3 - sm);
                   d5 = fabs(d5 - sm);
```

```
                d6 = fabs(d6 - sm);
                if(d3<st && d5<st && d6<st && s3 == s5 && s3 == s6){
                  jj = 0;                                  /* d3=d5=d6 SP2 */
                  for(n=(j-Ext1);n<=(j+Ext1+1);n++){
                    if(sinal2==1){
                      qtofsc[R] = oitofsc[n]*temp1[jj] + qtofsc[R];
                      }
                      else{
                       qtofsc[R] = oitofsc[n]*temp3[jj] + qtofsc[R];
                       }
                    jj++;
                    }
                sinal2 = (sinal2==1)?2:1;
                }
                else{
                  jj = 0;
                  for(n=(j-Ext);n<=(j+Ext+1);n++){
                    if(sinal2==1){
                       qtofsc[R] = oitofsc[n]*temp2[jj] + qtofsc[R];
                       }
                       else{
                        qtofsc[R] = oitofsc[n]*temp4[jj] + qtofsc[R];
                        }
                    jj++;
                    }
                    sinal2 = (sinal2==1)?2:1;
                    if(s1*s2>0.0){sinal=1.0;} else{sinal=-1.0;}
                    qtofsc[R]=qtofsc[R]+sinal*B*(oitofsc[j]-oitofsc[j+1]);
                  } /* conforme sinal decrementa ou incrementa fator     */
                    /* de correcao                                       */
                }
              }
              *cc = faprox(qtofsc[R]);  /* Transforma real para inteiro    */
             /* Soma do valor quadratico e do erro quadratico para 484 valores d
a linha */
              qtofsc[R] = 0.0;
              fprintf(saidqto,"%4d",*cc);    /* Imprime a saida de 4fsc    */
              } /* for R */
        } /* for I */
        fclose(saidqto); fclose(entoito);
       } /* else */
 }


/* Funcao sa(), calcula para cada contribuicao do somatorio, o valor da    */
/* sample correspondente                                                   */

void sa(Extt,tempp,k)
     int Extt,k;
     float *tempp;
     {
     int n1 = 1;
     int n2 = -Extt;
     float kk = 0.0;
     float spc = 0.0;          /* Acumulado da funcao sample()             */
     float Er  = 0.0;          /* Erro do valor acumulado da funcao sample */
     while(n2<=Extt+1){
        if(k == 1){
          kk = ((2.0/3.0)*PI - n2*PI);
          }
          else{
```

```
            kk = ((2.0/6.0)*PI - n2*PI);
            }
        tempp[n1-1] = (sin(kk))/kk;
        spc       = tempp[n1-1] + spc;
        n1++;
        n2++;
        }
        Er = 1.0 - spc;          /* Efetua a correcao das contribuicoes  ...*/
        for(n1=1;n1<=2*Extt+2;n1++){ /* da sample atraves da media ponderada */
            tempp[n1-1] = tempp[n1-1]*Er/spc + tempp[n1-1];
            }
    }


/* Funcao faprox(), aproximacao para inteiro mais proximo                    */

int faprox(r)
   float r;
   {
   float rf;
   int   ri;
   rf = 0.5 + r;
   ri = floor((float)rf);
        return(ri);
   }
```

```
/* Programa dezqtr.c                                                    */
/* Le um arquivo na frequencia de 10Mhz 512X512 transforma para 8/3fsc  */
/* (504X512 amostras ativas em ASC) e                                   */
/* converte o mesmo para  4fsc (755X512 amostras ativas em ASC).        */
/* Usa a media ponderada acrescida aos coeficientes da funcao sample()  */

#include <stdio.h>
#include <math.h>
#define PI 3.1415927
#define Colqto  755          /* 755 Numero de colunas do sinal em 4fsc    */
#define Coloit  503          /* 511 p/ 10Mhz, 503 Num.   de colunas do si */
#define Coloite 511          /* nal em 8/3fsc  */
#define Linhas  512       /* 512 Num. de linhas dos sinais em 4fsc e 10Mhz */
#define Ext      2           /* Num. de contribuicoes para o somatorio da */
#define Ext1     1           /* Sample. "n" varia de -Ext a +Ext+1;       */
#define st       1           /* delta, variacao para confirmacao de sinal */
#define zero     0
float oitofsc[Coloit+2*Ext+1];  /* Amostras ativas reais 10Mhz, mais Ext   */
                            /* amostras acrescidas aos extremos para cor */
                            /* recao das primeiras amostras de 4fsc      */
float qtofsc[Colqto];       /* Amostras ativas reais em 4fsc, uma linha  */
float temp1[2*Ext1 + 2];    /* Coeficientes da funcao Sample()           */
float temp2[2*Ext+2];
float temp3[2*Ext1+2];
float temp4[2*Ext+2];
void sa();
int faprox();

main (argc,argv)
   int argc;
   char *argv[];
   {
   int e,n1,j,jj,nzeros,k,Exttemp,difint,sinal2;
   float N;                 /* Posicao da R-esima amostra de 4fsc em 10Mhz */
   int n;                   /* n-esima amostra ativa de 10Mhz  0<=n<=503   */
   int l;                   /* Numero de linhas em 10Mhz e 4fsc 1<=l<=512  */
   int R;                   /* R-esima amostra ativa de 4fsc, 0<=R<=754    */
   float kk;
   float B;                 /* Fator de correcao conforme Ext              */
   float Extind = 0.0;      /* Fator de correcao dependente de Ext         */
   float s1, s2, sinal;     /* Identificam sinal de incremento/decremento  */
   float s3, s4, s5, s6, s7,sm;   /* Identificam tipo de sinal, se tende   */
   float d3, d4, d5, d6, d7;      /* a senoide, constante ou triangular    */
   FILE *entoito;   /* Arquivo de entrada 10Mhz, 503X512(conver.) amostras */
   FILE *saidqto;           /* Arquivo de saida 4fsc, 755X512 amostras     */
   int  *cc;                /* Inteiro correspondente ao sinal 4fsc        */
   cc = (int*)malloc(sizeof(int));   /* Espaco de memoria alocada para cc  */
     if(argc!=3){                    /* Se o argumento for diferente de 3, */
                                     /* imprime a forma de uso na tela     */
     printf("\nUso: dezqtr Orig_10Mhz Pro_4fsc ");
     exit(1);
     }
     if((entoito = fopen(argv[1],"r")) == NULL){
       printf("Erro na abertura do arquivo de entrada:%s\n", argv[1]);
       exit(1);
       }
     if((saidqto = fopen(argv[2],"wa")) == NULL){
       printf("Erro na abertura do arquivo de saida: %s\n",argv[2]);
       exit(1);
       }
```

```
      else{
        B = Extind/127;
        for(k=0;k<=(Coloit+2*Ext);k++){ oitofsc[k] = 0.0; }
        for(k=0;k<=(Colqto-1);k++){ qtofsc[k] = 0.0;}
        for(k=0;k<=2*Ext+1;k++){ temp2[k] = 0.0; temp4[k] = 0.0;}
        for(k=0;k<=2*Ext1+1;k++){ temp1[k] = 0.0; temp3[k] = 0.0; }
        sa(Ext,temp2,1);
        sa(Ext,temp4,2);
        sa(Ext1,temp1,1);
        sa(Ext1,temp3,2);
        for(l=1;l<=Linhas;l++){              /* Para todas as linhas de 8/3fs */
         sinal2 = 1;
         for(n=0;n<=(Coloite);n++){ /* Le uma linha do sinal 8/3fsc  */
           fscanf(entoito,"%4d",&e);
              if(n>=4){
                if(n<=507){
                   oitofsc[n-1] = (float)e;
                   }
                }
           }
         for(k=0;k<=(Ext-1);k++){              /* Preenche as Ext amostras com */
           oitofsc[k] = oitofsc[Ext+1];        /* a segunda amostra de 8/3fsc  */
           }
         for(k=Coloit+Ext;k<=Coloit+2*Ext-1;k++){ /* Preenche as Ext amostras*/
           oitofsc[k] = oitofsc[Ext-2];    /* finais com a penultima de 8/3fsc*/
           }
         for(R=0;R<=(Colqto-1);R++){       /* Calculo das amostras de 4fsc   */
          N = (float)R/1.5 + Ext;                  /* N-esima amostra de 8/3fsc.
  */
          j = N;                           /* R, posicao da amostra em 4fsc  */
          if(fmod(N,(float)j)==0){
             qtofsc[R] = oitofsc[j];       /* Amostragem coincide            */
             }
             else{                         /* Calculo atraves da sample()    */

/* Verifica o tipo de sinal p/ incrementar/decrementar o fator de correcao */

               s1 = oitofsc[j] - oitofsc[j-1];
               s2 = oitofsc[j-1] - oitofsc[j+2];
               d3 = s1;
               if(d3>0.0){s3=1.0;} else{s3=-1.0;}
               d4 = oitofsc[j-1] - oitofsc[j-2];
               /* if(d4>0.0){s4=1.0;} else{s4=-1.0;} */
               d5 = oitofsc[j+1] - oitofsc[j]  ;
               if(d5>0.0){s5=1.0;} else{s5=-1.0;}
               d6 = oitofsc[j+2] - oitofsc[j+1];
               if(d6>0.0){s6=1.0;} else{s6=-1.0;}
               d7 = oitofsc[j+3] - oitofsc[j+2];
               /* if(d7>0.0){s7=1.0;} else{s7=-1.0;} */
               nzeros = 0;
               d3 = fabs(d3); if(d3 == zero){nzeros++;}
               d4 = fabs(d4); if(d4 == zero){nzeros++;}
               d5 = fabs(d5); if(d5 == zero){nzeros++;}
               d6 = fabs(d6); if(d6 == zero){nzeros++;}
               d7 = fabs(d7); if(d7 == zero){nzeros++;}
               if(nzeros >= 3){  /* d1=d2=d4=d5=0, Bordas, regioes uniformes */
                 if(sinal2==1){
                   qtofsc[R] = (2.0/6.0)*oitofsc[j] + (2.0/3.0)*oitofsc[j+1];
                   }
                   else{
                    qtofsc[R] = (2.0/3.0)*oitofsc[j] + (2.0/6.0)*oitofsc[j+1];
```

```
                }
              sinal2 = (sinal2==1)?2:1;
              }
              else{
                sm = (d3 + d5 + d6)/3.0;
                d3 = fabs(d3 - sm);
                d5 = fabs(d5 - sm);
                d6 = fabs(d6 - sm);
                if(d3<st && d5<st && d6<st && s3 == s5 && s3 == s6){
                  jj = 0;                                   /* d3=d5=d6 SP2 */
                  for(n=(j-Ext1);n<=(j+Ext1+1);n++){
                    if(sinal2==1){
                      qtofsc[R] = oitofsc[n]*temp1[jj] + qtofsc[R];
                      }
                      else{
                       qtofsc[R] = oitofsc[n]*temp3[jj] + qtofsc[R];
                       }
                    jj++;
                    }
                  sinal2 = (sinal2==1)?2:1;
                  }
                  else{
                    jj = 0;
                    for(n=(j-Ext);n<=(j+Ext+1);n++){
                      if(sinal2==1){
                         qtofsc[R] = oitofsc[n]*temp2[jj] + qtofsc[R];
                         }
                         else{
                          qtofsc[R] = oitofsc[n]*temp4[jj] + qtofsc[R];
                          }
                      jj++;
                      }
                      sinal2 = (sinal2==1)?2:1;
                      if(s1*s2>0.0){sinal=1.0;} else{sinal=-1.0;}
                      qtofsc[R]=qtofsc[R]+sinal*B*(oitofsc[j]-oitofsc[j+1]);
                    } /* conforme sinal decrementa ou incrementa fator      */
                      /* de correcao                                        */
                }
             }
             *cc = faprox(qtofsc[R]);  /* Transforma real para inteiro     */
            /* Soma do valor quadratico e do erro quadratico para 484 valores d
a linha */
             qtofsc[R] = 0.0;
             fprintf(saidqto,"%4d",*cc);     /* Imprime a saida de 4fsc     */
             } /* for R */
        } /* for I */
        fclose(saidqto); fclose(entoito);
       } /* else */
 }


/* Funcao sa(), calcula para cada contribuicao do somatorio, o valor da     */
/* sample correspondente                                                    */

void sa(Extt,tempp,k)
     int Extt,k;
     float *tempp;
     {
     int n1 = 1;
     int n2 = -Extt;
     float kk = 0.0;
```

```
      float spc = 0.0;          /* Acumulado da funcao sample()             */
      float Er  = 0.0;          /* Erro do valor acumulado da funcao sample */
      while(n2<=Extt+1){
         if(k == 1){
           kk = ((2.0/3.0)*PI - n2*PI);
           }
           else{
             kk = ((2.0/6.0)*PI - n2*PI);
             }
         tempp[n1-1] = (sin(kk))/kk;
         spc       = tempp[n1-1] + spc;
         n1++;
         n2++;
         }
         Er = 1.0 - spc;          /* Efetua a correcao das contribuicoes  ...*/
         for(n1=1;n1<=2*Extt+2;n1++){ /* da sample atraves da media ponderada */
             tempp[n1-1] = tempp[n1-1]*Er/spc + tempp[n1-1];
             }
      }


/* Funcao faprox(), aproximacao para inteiro mais proximo                   */

int faprox(r)
    float r;
   {
   float rf;
   int   ri;
   rf = 0.5 + r;
   ri = floor((float)rf);
        return(ri);
   }
```

```
# This file was generated by 'gxv' from 'sdviP.G'.

# Parameters.

PROGRAM = sdviP
SOURCES.c = sdviP.c sdvipro.c sdvii.c sdviv.c sdvia.c FSProc.c rawtopgm.c rgb3to
ppm.c binasc.c ascbin.c rgbm.c mrgb.c binasc83.c ascbin83.c rgbm83.c mrgb83.c po
pupxview.c putimagepopup.c  iconimagepopup.c vxviewpopup.c viewimagepopup.c qtro
itos.c oitoqtrs.c dezqtrs.c
SOURCES.h =  FSProc.h  rawtopgm.h rgb3toppm.h binasc.h ascbin.h rgbm.h mrgb.h bi
nasc83.h ascbin83.h rgbm83.h mrgb83.h popupxview.h putimagepopup.h iconimagepopu
p.h vxviewpopup.h viewimagepopup.h qtroito.h oitoqtr.h dezqtr.h
SOURCES.G = sdviP.G sdvipro.G sdvii.G sdviv.G sdvia.G FSProc.G binasc83.G mrgb.G
 rgbm83.G ascbin83.G rgbm.G rawtopgm.G mrgb83.G binasc.G ascbin.G rgb3toppm.G po
pupxview.G viewimagepopup.G vxviewpopup.G iconimagepopup.G putimagepopup.G qtroi
to.G oitoqtr.G dezqtr.G
STUBS.G =

# Derived parameters.

SOURCES = \
        ${SOURCES.G} \
        ${SOURCES.h} \
        ${SOURCES.l} \
        ${SOURCES.y} \
        ${SOURCES.c}

TARGETS.c = \
        ${SOURCES.G:%.G=%_ui.c} \
        ${STUBS.G:%.G=%_stubs.c}

TARGETS.h = \
        ${SOURCES.G:%.G=%_ui.h} \
        ${SOURCES.l:%.l=%.h} \
        ${SOURCES.y:%.y=%.h}

TARGETS.other = \
        ${SOURCES.G:%.G=%.info}

TARGETS = \
        ${TARGETS.other} \
        ${TARGETS.h} \
        ${TARGETS.c}

OBJECTS = \
        ${SOURCES.c:%.c=%.o} \
        ${TARGETS.c:%.c=%.o}

# Compiler flags.

CFLAGS += -g
CPPFLAGS += -I${GUIDEHOME}/include -I${OPENWINHOME}/include
LDFLAGS += -L${GUIDEHOME}/lib -L${OPENWINHOME}/lib
LDLIBS += -lguide -lguidexv -lxview -lolgx -lX

# Standard targets.

all:    ${TARGETS.other} ${PROGRAM}
objects: ${SOURCES.c} ${TARGETS.c} ${TARGETS.h} ${OBJECTS}
sources: ${SOURCES}
```



```
targets: ${SOURCES} ${TARGETS}

${PROGRAM}: ${SOURCES.c} ${TARGETS.c} ${TARGETS.h} ${OBJECTS}
        ${LINK.c} -o ${PROGRAM} ${OBJECTS} ${LDLIBS}

# Targets to be used by a certain CASE tool.

stest: ${SOURCES.c} ${TARGETS.c} ${TARGETS.h} ${OBJECTS}
        #load ${CFLAGS} ${CPPFLAGS} ${LDFLAGS} ${TARGET_ARCH} \
                -o ${PROGRAM} ${SOURCES.c} ${TARGETS.c} ${LDLIBS}

%_ui.c + %_ui.h + %_stubs.c + %.info: %.G
        ${GUIDEHOME}/bin/gxv $*

clean:
        ${RM} ${TARGETS} ${OBJECTS} ${SOURCES.G:%.G=%_stubs.c} core
```

```
/*
 * sdviP_stubs.c - Notify and event callback function stubs.
 * This file was generated by 'gxv' from 'sdviP.G'.
 */

#include <stdio.h>
#include <sys/param.h>
#include <sys/types.h>
#include <xview/xview.h>
#include <xview/panel.h>
#include <xview/textsw.h>
#include <xview/xv_xrect.h>
#include <gdd.h>
#include "sdviP_ui.h"
#include "sdvii_ui.h"     /* Define  o objeto  sdvii_informacoes          */
#include "sdvia_ui.h"     /* Define  o objeto  Sdvia_aquisicao            */
#include "sdviv_ui.h"     /* Define  o objeto  sdviv_visualizacao         */
#include "sdvipro_ui.h"   /* Define  o objeto  sdvipro_processamento      */
#include "FSProc_ui.h"    /* Define  o objeto  FSProc_FSProcpoup          */
#include "rawtopgm_ui.h"  /* Define  o objeto  rawtopgm_popup1rawtopgm    */
#include "rgb3toppm_ui.h" /* Define  o objeto  rgb3toppm_popup1rgb3toppm  */
#include "binasc_ui.h"    /* Define  o objeto  binasc_popupbinasc         */
#include "rgbm_ui.h"      /* Define  o objeto  rgbm_popup1rgbm            */
#include "mrgb_ui.h"      /* Define  o objeto mrgb_popup1mrgb             */
#include "ascbin_ui.h"    /* Define  o objeto ascbin_popupascbin          */
#include "binasc83_ui.h"  /* Define  o objeto binasc83_popupbinasc83      */
#include "rgbm83_ui.h"    /* Define  o objeto rgbm83_popup1rgbm83         */
#include "mrgb83_ui.h"    /* Define  o objeto mrgb83_popup1_objects       */
#include "ascbin83_ui.h"      /* Define  o objeto ascbin83_popupascbin83
*/
#include "popupxview_ui.h"    /* Define o objeto popupxview_popupxview
*/
#include "putimagepopup_ui.h" /* Define o objeto putimagepopup_putimagepopup
*/
#include "iconimagepopup_ui.h" /* Define o objeto iconimagepopup_iconimagepopup
*/
#include "viewimagepopup_ui.h" /* Define o objeto viewimagepopup_viewimagepopup
*/
#include "vxviewpopup_ui.h"   /* Define  o objeto vxviewpopup_vxviewpopup
*/
#include "oitoqtr_ui.h"       /* Define o objeto oitoqtr_popupoitoqtr
*/
#include "qtroito_ui.h"       /* Define o objeto qtroito_popup1
*/
#include "dezqtr_ui.h"        /* Define o objeto dezqtr_popup1
*/


/*
 * Global object definitions.
 */
sdviP_window1_objects   *SdviP_window1;

sdvia_aquisicao_objects              *Sdvia_aquisicao;
sdvii_informacoes_objects            *sdvii_informacoes;
sdvipro_processamento_objects        *Sdvipro_processamento;
sdviv_visualizacao_objects           *sdviv_visualizacao;
FSProc_FSProcpoup_objects            *FSProc_FSProcpoup;
rawtopgm_popup1rawtopgm_objects      *Rawtopgm_popup1rawtopgm;
rgb3toppm_popup1rgb3toppm_objects    *Rgb3toppm_popup1rgb3toppm;
```



```
binasc_popupbinasc_objects            *Binasc_popupbinasc;
rgbm_popup1rgbm_objects               *Rgbm_popup1rgbm;
mrgb_popup1mrgb_objects               *Mrgb_popup1mrgb;
ascbin_popupascbin_objects            *Ascbin_popupascbin;
binasc83_popupbinasc83_objects        *Binasc83_popupbinasc83;
rgbm83_popup1rgbm83_objects           *Rgbm83_popup1rgbm83;
mrgb83_popup1_objects                 *Mrgb83_popup1;
ascbin83_popupascbin83_objects        *Ascbin83_popupascbin83;
popupxview_popupxview_objects         *Popupxview_popupxview;
putimagepopup_putimagepopup_objects   *Putimagepopup_putimagepopup;
iconimagepopup_iconimagepopup_objects *Iconimagepopup_iconimagepopup;
viewimagepopup_viewimagepopup_objects *Viewimagepopup_viewimagepopup;
vxviewpopup_vxviewpopup_objects       *Vxviewpopup_vxviewpopup;
oitoqtr_popupoitoqtr_objects          *Oitoqtr_popupoitoqtr;
qtroito_popup1_objects                *Qtroito_popup1;
dezqtr_popup1_objects                 *Dezqtr_popup1;


/* #ifdef MAIN  Retirada para nao criar mu'ltiplas Mains */

/*
 * Instance XV_KEY_DATA key.  An instance is a set of related
 * user interface objects.  A pointer to an object's instance
 * is stored under this key in every object.  This must be a
 * global variable.
 */
Attr_attribute  INSTANCE;

main(argc, argv)
        int     argc;
        char    **argv;
{

        /*
         * Initialize XView.
         */
        xv_init(XV_INIT_ARGC_PTR_ARGV, &argc, argv, NULL);
        INSTANCE = xv_unique_key();

        /*
         * Initialize user interface components.
         * Do NOT edit the object initializations by hand.
         */
        SdviP_window1 = sdviP_window1_objects_initialize(NULL, NULL);     '

           /* Incializa o processamento da Popup PROCESSAMENTO - SDVI   */
        Sdvipro_processamento = sdvipro_processamento_objects_initialize(NULL, S
dviP_window1->window1);

           /* Incializa o processamento da Popup INFORMACOES  -  SDVI   */
        sdvii_informacoes = sdvii_informacoes_objects_initialize(NULL, SdviP_win
dow1->window1);

           /* Incializa o processamento da Popup  VISUALIZACAO  -  SDVI */
        sdviv_visualizacao = sdviv_visualizacao_objects_initialize(NULL, SdviP_w
indow1->window1);

           /* Incializa o processamento da Popup AQUISICAO - SDVI       */

        Sdvia_aquisicao = sdvia_aquisicao_objects_initialize(NULL, SdviP_window1
```

```
->window1);
            /* Inicializa o processamento da Popup FSProc                */
  FSProc_FSProcpoup = FSProc_FSProcpoup_objects_initialize(NULL, SdviP_window1->
window1);
           /* Incializa o processamento da Popup rawtopgm                */
         Rawtopgm_popup1rawtopgm = rawtopgm_popup1rawtopgm_objects_initialize(NUL
L, SdviP_window1->window1);
           /* Incializa o processamento da Popup rgb3toppm               */
         Rgb3toppm_popup1rgb3toppm = rgb3toppm_popup1rgb3toppm_objects_initialize
(NULL, SdviP_window1->window1);
           /* Incializa o processamento da Popup binasc                  */
         Binasc_popupbinasc = binasc_popupbinasc_objects_initialize(NULL, SdviP_w
indow1->window1);
           /* Incializa o processamento da Popup ascbin                  */
         Ascbin_popupascbin = ascbin_popupascbin_objects_initialize(NULL, SdviP_w
indow1->window1);
           /* Incializa o processamento da Popup rgbm                    */
         Rgbm_popup1rgbm = rgbm_popup1rgbm_objects_initialize(NULL, SdviP_window1
->window1);
           /* Incializa o processamento da Popup mrgb                    */
         Mrgb_popup1mrgb = mrgb_popup1mrgb_objects_initialize(NULL, SdviP_window1
->window1);

           /* Incializa o processamento da Popup binasc83                */
         Binasc83_popupbinasc83 = binasc83_popupbinasc83_objects_initialize(NULL,
 SdviP_window1->window1);

           /* Incializa o processamento da Popup ascbin83                */
         Ascbin83_popupascbin83 = ascbin83_popupascbin83_objects_initialize(NULL,
  SdviP_window1->window1);

           /* Incializa o processamento da Popup rgbm83                  */
         Rgbm83_popup1rgbm83 = rgbm83_popup1rgbm83_objects_initialize(NULL, SdviP
_window1->window1);

           /* Incializa o processamento da Popup mrgb83                  */
         Mrgb83_popup1 = mrgb83_popup1_objects_initialize(NULL, SdviP_window1->wi
ndow1);


           /* Incializa o processamento da Popup popupxview              */
         Popupxview_popupxview = popupxview_popupxview_objects_initialize(NULL, S
dviP_window1->window1);

          /* Incializa o processamento da Popup putimagepopup           */
         Putimagepopup_putimagepopup = putimagepopup_putimagepopup_objects_initia
lize(NULL, SdviP_window1->window1);

         /* Incializa o processamento da Popup   Iconimagepopup         */
         Iconimagepopup_iconimagepopup = iconimagepopup_iconimagepopup_objects_in
itialize(NULL, SdviP_window1->window1);

         /* Incializa o processamento da Popup Viewimagepopup           */
         Viewimagepopup_viewimagepopup = viewimagepopup_viewimagepopup_objects_in
itialize(NULL, SdviP_window1->window1);

         /* Incializa o processamento da Popup vxviewpopup              */
         Vxviewpopup_vxviewpopup = vxviewpopup_vxviewpopup_objects_initialize(NUL
L, SdviP_window1->window1);

       /* Inicializa o processamento da Popup oitoqtr                   */
```



```
         Oitoqtr_popupoitoqtr = oitoqtr_popupoitoqtr_objects_initialize(NULL,Sdvi
P_window1->window1);

        /* Inicializa Popup qtroito     */
         Qtroito_popup1 = qtroito_popup1_objects_initialize(NULL,SdviP_window1->w
indow1);

         /* Inicializa Popup dezqtr     */
         Dezqtr_popup1 = dezqtr_popup1_objects_initialize(NULL,SdviP_window1->win
dow1);




         /*
          * Turn control over to XView.
          */
         xv_main_loop(SdviP_window1->window1);
         exit(0);

}

/* #endif   Retirada */


/*
 * Notify callback function for 'setting1'.
 */
void
processamento(item, value, event)
         Panel_item      item;
         int             value;
         Event           *event;
{
         sdviP_window1_objects *ip = (sdviP_window1_objects *) xv_get(item, XV_KE
Y_DATA, INSTANCE);
        if(value ==0)
              xv_set(Sdvia_aquisicao->aquisicao, XV_SHOW, TRUE, 0);
        if(value ==1)
              xv_set(Sdvipro_processamento->processamento, XV_SHOW, TRUE, 0);
        if(value ==2)
              xv_set(sdviv_visualizacao->visualizacao, XV_SHOW, TRUE, 0); '
        if(value ==3)
              xv_set(sdvii_informacoes->informacoes, XV_SHOW, TRUE, 0);

         /* fprintf(stderr, "sdviP: processamento: value: %u\n", value); ret */

         /* gxv_start_connections DO NOT EDIT THIS SECTION */

         /* gxv_end_connections */

}
```

```
/*
 * FSProc_stubs.c - Notify and event callback function stubs.
 * This file was generated by 'gxv' from 'FSProc.G'.
 */

#include <stdio.h>
#include <sys/param.h>
#include <sys/types.h>
#include <xview/xview.h>
#include <xview/panel.h>
#include <xview/textsw.h>
#include <xview/xv_xrect.h>
#include <xview/notice.h>     /* inclui'do para acesso de notice_prompt() */
#include "FSProc_ui.h"

#include "rawtopgm_ui.h"   /* Define o objeto  rawtopgm_popup1rawtopgm    */
#include "rgb3toppm_ui.h"  /* Define o objeto  rgb3toppm_popup1rgb3toppm  */
#include "binasc_ui.h"     /* Define o objeto  binasc_popupbinasc_objects */
#include "rgbm_ui.h"       /* Define o objeto  rgbm_popup1rgbm            */
#include "mrgb_ui.h"       /* Define o objeto mrgb_popup1mrgb             */
#include "ascbin_ui.h"     /* Define o objeto ascbin_popupascbin          */
#include "binasc83_ui.h"   /* Define o objeto binasc83_popupbinasc83      */
#include "rgbm83_ui.h"     /* Define o objeto rgbm83_popup1rgbm83         */
#include "mrgb83_ui.h"     /* Define o objeto mrgb83_popup1_objects       */
#include "ascbin83_ui.h"   /* Define o objeto ascbin83_popupascbin83      */
#include "oitoqtr_ui.h"    /* Define o objeto oitoqtr_popupoitoqtr        */
#include "qtroito_ui.h"    /* Define o objeto qtroito_popup1              */
#include "dezqtr_ui.h"     /* Define o objeto dezqtr_popup1               */


void    GiftoVif();
void    ViftoGif();
void    JpegtoVif();
void    ViftoJpeg();
void    BytetoVif();
void    ViftoByte();

/*
 * Declara,cao de varia'veis para as Popup's a serem chamadas
 */
rawtopgm_popup1rawtopgm_objects       *Rawtopgm_popup1rawtopgm;
rgb3toppm_popup1rgb3toppm_objects     *Rgb3toppm_popup1rgb3toppm;
binasc_popupbinasc_objects            *Binasc_popupbinasc;
rgbm_popup1rgbm_objects               *Rgbm_popup1rgbm;
mrgb_popup1mrgb_objects               *Mrgb_popup1mrgb;
ascbin_popupascbin_objects            *Ascbin_popupascbin;
binasc83_popupbinasc83_objects        *Binasc83_popupbinasc83;
rgbm83_popup1rgbm83_objects           *Rgbm83_popup1rgbm83;
mrgb83_popup1_objects                 *Mrgb83_popup1;
ascbin83_popupascbin83_objects        *Ascbin83_popupascbin83;
oitoqtr_popupoitoqtr_objects          *Oitoqtr_popupoitoqtr;
qtroito_popup1_objects                *Qtroito_popup1;
dezqtr_popup1_objects                 *Dezqtr_popup1;



/*
 * Global object definitions.
 */
FSProc_FSProcpoup_objects       *FSProc_FSProcpoup;
```



```
#ifdef MAIN

/*
 * Instance XV_KEY_DATA key.  An instance is a set of related
 * user interface objects.  A pointer to an object's instance
 * is stored under this key in every object.  This must be a
 * global variable.
 */
Attr_attribute  INSTANCE;

main(argc, argv)
        int     argc;
        char    **argv;
{
        /*
         * Initialize XView.
         */
        xv_init(XV_INIT_ARGC_PTR_ARGV, &argc, argv, NULL);
        INSTANCE = xv_unique_key();

        /*
         * Initialize user interface components.
         * Do NOT edit the object initializations by hand.
         */
        FSProc_FSProcpoup = FSProc_FSProcpoup_objects_initialize(NULL, NULL);

        /* Inicializa Popup mrgb83     */
        Mrgb83_popup1 = mrgb83_popup1_objects_initialize(NULL, FSProc_FSProcpoup
->FSProcpoup);
        /* Inicializa Popup rgbm83     */
        Rgbm83_popup1rgbm83 = rgbm83_popup1rgbm83_objects_initialize(NULL, FSPro
c_FSProcpoup->FSProcpoup);
        /* Inicializa Popup ascbin83     */
        Ascbin83_popupascbin83 = ascbin83_popupascbin83_objects_initialize(NULL,
FSProc_FSProcpoup->FSProcpoup);
        /* Inicializa Popup binasc83     */
Binasc83_popupbinasc83 = binasc83_popupbinasc83_objects_initialize(NULL, FSProc_
FSProcpoup->FSProcpoup);
        /* Inicializa Popup mrgb     */
Mrgb_popup1mrgb = mrgb_popup1mrgb_objects_initialize(NULL, FSProc_FSProcpoup->FS
Procpoup);
        /* Inicializa Popup rgbm     */
        Rgbm_popup1rgbm = rgbm_popup1rgbm_objects_initialize(NULL, FSProc_FSProc
poup->FSProcpoup);
        /* Inicializa Popup ascbin     */
        Ascbin_popupascbin = ascbin_popupascbin_objects_initialize(NULL, FSProc_
FSProcpoup->FSProcpoup);
        /* Inicializa Popup binasc     */
        Binasc_popupbinasc = binasc_popupbinasc_objects_initialize(NULL, FSProc_
FSProcpoup->FSProcpoup);
        /* Inicializa Popup rgb3toppm     */
        Rgb3toppm_popup1rgb3toppm = rgb3toppm_popup1rgb3toppm_objects_initialize
(NULL,FSProc_FSProcpoup->FSProcpoup);
        /* Inicializa Popup rawtopgm     */
        Rawtopgm_popup1rawtopgm = rawtopgm_popup1rawtopgm_objects_initialize(NUL
L,FSProc_FSProcpoup->FSProcpoup);
        /* Inicializa Popup oitoqtr     */
        Oitoqtr_popupoitoqtr = oitoqtr_popupoitoqtr_objects_initialize(NULL,FSPr
oc_FSProcpoup->FSProcpoup );
        /* Inicializa Popup qtroito     */
```

```
        Qtroito_popup1 = qtroito_popup1_objects_initialize(NULL, FSProc_FSProcpo
up->FSProcpoup);
        /* Inicializa Popup dezqtr     */
        Dezqtr_popup1 = dezqtr_popup1_objects_initialize(NULL,FSProc_FSProcpoup-
>FSProcpoup);

        /*
         * Turn control over to XView.
         */
        xv_main_loop(FSProc_FSProcpoup->FSProcpoup);
        exit(0);
}

#endif


/*
 * Menu handler for 'FSQuatpara (10Mhz)'.
 */
Menu_item
QparaDez(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        FSProc_FSProcpoup_objects * ip = (FSProc_FSProcpoup_objects *) xv_get(it
em, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
         /* Usa o pacote notice para informar sobre a transforma,cao */

         result = notice_prompt(ip->FSProcpoup, NULL,
            /* NOTICE_FOCUS_XY,        event_x(event), event_y(event), */
             NOTICE_MESSAGE_STRING,  "Por anexar!",
             NOTICE_BUTTON_YES,      "OK",
             NULL);
          if (result == NOTICE_YES)
                fputs("FSProc: QparaDez: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'FSQuatpara (13.5Mhz)'.
 */
Menu_item
```



```
Qparatreze(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        FSProc_FSProcpoup_objects * ip = (FSProc_FSProcpoup_objects *) xv_get(it
em, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
           /* Usa o pacote notice para informar sobre a transforma,cao */

         result = notice_prompt(ip->FSProcpoup, NULL,
           /*  NOTICE_FOCUS_XY,        event_x(event), event_y(event), */
             NOTICE_MESSAGE_STRING,  "Por anexar!",
             NOTICE_BUTTON_YES,      "OK",
             NULL);
          if (result == NOTICE_YES)
                fputs("FSProc: Qparatreze: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'FSQuatpara (8/3fsc)'.
 */
Menu_item
QparaOito(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        FSProc_FSProcpoup_objects * ip = (FSProc_FSProcpoup_objects *) xv_get(it
em, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
             xv_set(Qtroito_popup1->popup1, XV_SHOW, TRUE, 0);

             fputs("FSProc: QparaOito: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */
```

```
                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'FSDezpara (4fsc)'.
 */
Menu_item
DezparaQ(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        FSProc_FSProcpoup_objects * ip = (FSProc_FSProcpoup_objects *) xv_get(it
em, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
                xv_set(Dezqtr_popup1->popup1, XV_SHOW, TRUE, 0);

                fputs("FSProc: DezparaQ: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'FSDezpara (13.5Mhz)'.
 */
Menu_item
DezparaTreze(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        FSProc_FSProcpoup_objects * ip = (FSProc_FSProcpoup_objects *) xv_get(it
em, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;
```



```
        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
          /* Usa o pacote notice para informar sobre a transforma,cao */
         result = notice_prompt(ip->FSProcpoup, NULL,
            /* NOTICE_FOCUS_XY,        event_x(event), event_y(event), */
             NOTICE_MESSAGE_STRING,  "Por anexar!",
             NOTICE_BUTTON_YES,      "OK",
             NULL);
          if (result == NOTICE_YES)
             fputs("FSProc: DezparaTreze: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'FSDezpara (8/3fsc)'.
 */
Menu_item
DezparaOito(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        FSProc_FSProcpoup_objects * ip = (FSProc_FSProcpoup_objects *) xv_get(it
em, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
          /* Usa o pacote notice para informar sobre a transforma,cao */'

         result = notice_prompt(ip->FSProcpoup, NULL,
            /* NOTICE_FOCUS_XY,        event_x(event), event_y(event), */
             NOTICE_MESSAGE_STRING,  "Por anexar!",
             NOTICE_BUTTON_YES,      "OK",
             NULL);
          if (result == NOTICE_YES)
            fputs("FSProc: DezparaOito: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
```

```
                break;
        }
        return item;
}

/*
 * Menu handler for 'FSOitterpara (4fsc)'.
 */
Menu_item
OitoparaQ(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        FSProc_FSProcpoup_objects * ip = (FSProc_FSProcpoup_objects *) xv_get(it
em, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
                xv_set(Oitoqtr_popupoitoqtr->popupoitoqtr, XV_SHOW, TRUE, 0);
               /* fputs("FSProc: OitoparaQ: MENU_NOTIFY\n", stderr); */

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'FSOitterpara (10Mhz)'.
 */
Menu_item
OitoparaDez(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        FSProc_FSProcpoup_objects * ip = (FSProc_FSProcpoup_objects *) xv_get(it
em, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
           /* Usa o pacote notice para informar sobre a transforma,cao */

         result = notice_prompt(ip->FSProcpoup, NULL,
```



```
            /* NOTICE_FOCUS_XY,        event_x(event), event_y(event), */
             NOTICE_MESSAGE_STRING,  "Por anexar!",
             NOTICE_BUTTON_YES,      "OK",
             NULL);
          if (result == NOTICE_YES)
             fputs("FSProc: OitoparaDez: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'FSOitterpara (13.5Mhz)'.
 */
Menu_item
OitoparaTreze(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        FSProc_FSProcpoup_objects * ip = (FSProc_FSProcpoup_objects *) xv_get(it
em, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
          /* Usa o pacote notice para informar sobre a transforma,cao */
             result = notice_prompt(ip->FSProcpoup, NULL,
            /* NOTICE_FOCUS_XY,        event_x(event), event_y(event), */
             NOTICE_MESSAGE_STRING,  "Por anexar!",
             NOTICE_BUTTON_YES,      "OK",
             NULL);
          if (result == NOTICE_YES)
             fputs("FSProc: OitoparaTreze: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'FSTrezpara (4fsc)'.
```

```
 */
Menu_item
TrezeparaQ(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        FSProc_FSProcpoup_objects * ip = (FSProc_FSProcpoup_objects *) xv_get(it
em, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
            /* Usa o pacote notice para informar sobre a transforma,cao */
             result = notice_prompt(ip->FSProcpoup, NULL,
            /* NOTICE_FOCUS_XY,        event_x(event), event_y(event), */
             NOTICE_MESSAGE_STRING,  "Por anexar!",
             NOTICE_BUTTON_YES,      "OK",
             NULL);
          if (result == NOTICE_YES)
             fputs("FSProc: TrezeparaQ: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'FSTrezpara (10Mhz)'.
 */
Menu_item
TrezeparaDez(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        FSProc_FSProcpoup_objects * ip = (FSProc_FSProcpoup_objects *) xv_get(it
em, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
           /* Usa o pacote notice para informar sobre a transforma,cao */
             result = notice_prompt(ip->FSProcpoup, NULL,
            /* NOTICE_FOCUS_XY,        event_x(event), event_y(event), */
             NOTICE_MESSAGE_STRING,  "Por anexar!",
```



```
             NOTICE_BUTTON_YES,      "OK",
             NULL);
          if (result == NOTICE_YES)
             fputs("FSProc: TrezeparaDez: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'FSTrezpara (8/3fsc)'.
 */
Menu_item
TrezeparaOito(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        FSProc_FSProcpoup_objects * ip = (FSProc_FSProcpoup_objects *) xv_get(it
em, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
           /* Usa o pacote notice para informar sobre a transforma,cao */
             result = notice_prompt(ip->FSProcpoup, NULL,
            /* NOTICE_FOCUS_XY,        event_x(event), event_y(event), */
             NOTICE_MESSAGE_STRING,  "Por anexar!",
             NOTICE_BUTTON_YES,      "OK",
             NULL);
          if (result == NOTICE_YES)
             fputs("FSProc: TrezeparaOito: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Notify callback function for 'buttonrawtopgm'.
 */
void
```

```
rawparapgm(item, event)
        Panel_item      item;
        Event           *event;
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);

        fputs("FSProc: rawparapgm\n", stderr);
        /* Chama Popup rawtopgm                         */
      xv_set(Rawtopgm_popup1rawtopgm->popup1rawtopgm, XV_SHOW, TRUE, 0);

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        /* gxv_end_connections */

}

/*
 * Notify callback function for 'buttonrgb3toppm'.
 */
void
rgbparappm(item, event)
        Panel_item      item;
        Event           *event;
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);

        /* fputs("FSProc: rgbparappm\n", stderr);  ret */
        /* Chama Popup rgb3toppm                        */
      xv_set(Rgb3toppm_popup1rgb3toppm->popup1rgb3toppm, XV_SHOW, TRUE, 0);

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        /* gxv_end_connections */

}

/*
 * Notify callback function for 'settingGifVif'.
 */
void
FSProc_FSProcpoup_settingGifVif_notify_callback(item, value, event)
        Panel_item      item;
        int             value;
        Event           *event;
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);

 /* Usa o pacote notice para informar sobre a transforma,cao */
         int      result;
         result = notice_prompt(ip->FSProcpoup, NULL,
             NOTICE_FOCUS_XY,        event_x(event), event_y(event),
             NOTICE_MESSAGE_STRING,  "Use o Khoros ou Xv",
             NOTICE_BUTTON_YES,      "OK",
             NULL);
          if (result == NOTICE_YES)
             return; /* Para voltar ao programa */

        /* fprintf(stderr, "FSProc: FSProc_FSProcpoup_settingGifVif_notify_callb
```



```
ack: value: %u\n", value); ret */

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        if (value == 0)
        {
                GiftoVif(item, value, event);
        }

        if (value == 1)
        {
                ViftoGif(item, value, event);
        }

        /* gxv_end_connections */

}

/*
 * User-defined action for 'settingGifVif'.
 */
void
GiftoVif(item, value, event)
        Panel_item      item;
        int             value;
        Event           *event;
{
        fputs("FSProc: GiftoVif\n", stderr);

}

/*
 * User-defined action for 'settingGifVif'.
 */
void
ViftoGif(item, value, event)
        Panel_item      item;
        int             value;
        Event           *event;
{
        fputs("FSProc: ViftoGif\n", stderr);

}

/*
 * Notify callback function for 'settingJpegVif'.
 */
void
FSProc_FSProcpoup_settingJpegVif_notify_callback(item, value, event)
        Panel_item      item;
        int             value;
        Event           *event;
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);

 /* Usa o pacote notice para informar sobre a transforma,cao */
         int      result;
         result = notice_prompt(ip->FSProcpoup, NULL,
             NOTICE_FOCUS_XY,        event_x(event), event_y(event),
             NOTICE_MESSAGE_STRING,  "Use o Khoros ou Xv",
```

```
            NOTICE_BUTTON_YES,      "OK",
            NULL);
          if (result == NOTICE_YES)
            return; /* Para voltar ao programa */

        /* fprintf(stderr, "FSProc: FSProc_FSProcpoup_settingJpegVif_notify_call
back: value: %u\n", value); ret */

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        if (value == 0)
        {
                JpegtoVif(item, value, event);
        }

        if (value == 1)
        {
                ViftoJpeg(item, value, event);
        }

        /* gxv_end_connections */

}

/*
 * User-defined action for 'settingJpegVif'.
 */
void
JpegtoVif(item, value, event)
        Panel_item      item;
        int             value;
        Event           *event;
{
        fputs("FSProc: JpegtoVif\n", stderr);
}

/*
 * User-defined action for 'settingJpegVif'.
 */
void
ViftoJpeg(item, value, event)
        Panel_item      item;
        int             value;
        Event           *event;
{
        fputs("FSProc: ViftoJpeg\n", stderr);
}

/*
 * Notify callback function for 'settingByteVif'.
 */
void
FSProc_FSProcpoup_settingByteVif_notify_callback(item, value, event)
        Panel_item      item;
        int             value;
        Event           *event;
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);
```



```
  /* Usa o pacote notice para informar sobre a transforma,cao */
        int      result;
        result = notice_prompt(ip->FSProcpoup, NULL,
            NOTICE_FOCUS_XY,        event_x(event), event_y(event),
            NOTICE_MESSAGE_STRING,  "Use o Khoros ou Xv",
            NOTICE_BUTTON_YES,      "OK",
            NULL);
         if (result == NOTICE_YES)
            return; /* Para voltar ao programa */


        /* fprintf(stderr, "FSProc: FSProc_FSProcpoup_settingByteVif_notify_call
back: value: %u\n", value); ret */

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        if (value == 0)
        {
                BytetoVif(item, value, event);
        }

        if (value == 1)
        {
                ViftoByte(item, value, event);
        }

        /* gxv_end_connections */

}

/*
 * User-defined action for 'settingByteVif'.
 */
void
BytetoVif(item, value, event)
        Panel_item      item;
        int             value;
        Event           *event;
{
        fputs("FSProc: BytetoVif\n", stderr);}

/*
 * User-defined action for 'settingByteVif'.
 */
void
ViftoByte(item, value, event)
        Panel_item      item;
        int             value;
        Event           *event;
{
        fputs("FSProc: ViftoByte\n", stderr);
}

/*
 * Notify callback function for 'buttonbinasc'.
 */
void
bintoasc(item, event)
        Panel_item      item;
        Event           *event;
```

```
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);

        /* fputs("FSProc: bintoasc\n", stderr); */
        /* Chama Popup rgb3toppm                */
      xv_set(Binasc_popupbinasc->popupbinasc, XV_SHOW, TRUE, 0);

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        /* gxv_end_connections */

}

/*
 * Notify callback function for 'buttonrgbm'.
 */
void
rgbtom(item, event)
        Panel_item      item;
        Event           *event;
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);

        fputs("FSProc: rgbtom\n", stderr);
         /* Chama Popup rgbm    */
         xv_set(Rgbm_popup1rgbm->popup1rgbm, XV_SHOW, TRUE, 0);

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        /* gxv_end_connections */

}

/*
 * Notify callback function for 'buttonmrgb'.
 */
void
mtorgb(item, event)
        Panel_item      item;
        Event           *event;
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);

        /* fputs("FSProc: mtorgb\n", stderr);      ret */
        /* Chama Popup mrgb                         */
        xv_set(Mrgb_popup1mrgb->popup1mrgb, XV_SHOW, TRUE, 0);

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        /* gxv_end_connections */

}

/*
 * Notify callback function for 'buttonascbin'.
 */
void
asctobin(item, event)
```



```
        Panel_item      item;
        Event           *event;
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);

        /* fputs("FSProc: asctobin\n", stderr); ret    */
        /* Chama Popup ascbin                          */
        xv_set(Ascbin_popupascbin->popupascbin, XV_SHOW, TRUE, 0);

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        /* gxv_end_connections */

}

/*
 * Notify callback function for 'buttonbinasc83'.
 */
void
bintoasc83(item, event)
        Panel_item      item;
        Event           *event;
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);

        /* fputs("FSProc: bintoasc83\n", stderr);  ret */
        /* Chama Popup binasc83                        */
        xv_set(Binasc83_popupbinasc83->popupbinasc83, XV_SHOW, TRUE, 0);

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        /* gxv_end_connections */

}

/*
 * Notify callback function for 'buttonrgbm83'.
 */
void
rgbtom83(item, event)
        Panel_item      item;
        Event           *event;
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);

        /* fputs("FSProc: rgbtom83\n", stderr); ret    */
        /* Chama Popup rgbm83                          */
        xv_set(Rgbm83_popup1rgbm83->popup1rgbm83, XV_SHOW, TRUE, 0);

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        /* gxv_end_connections */

}

/*
 * Notify callback function for 'buttonmrgb83'.
 */
```

```
void
mtorgb83(item, event)
        Panel_item      item;
        Event           *event;
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);

        /* fputs("FSProc: mtorgb83\n", stderr);  ret   */
        /* Chama Popup     mrgb83                      */
        xv_set(Mrgb83_popup1->popup1, XV_SHOW, TRUE, 0);

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        /* gxv_end_connections */

}

/*
 * Notify callback function for 'buttonascbin83'.
 */
void
asctobin83(item, event)
        Panel_item      item;
        Event           *event;
{
        FSProc_FSProcpoup_objects *ip = (FSProc_FSProcpoup_objects *) xv_get(ite
m, XV_KEY_DATA, INSTANCE);

        /* fputs("FSProc: asctobin83\n", stderr);  ret */
        /* Chama Popup ascbin83                        */
        xv_set(Ascbin83_popupascbin83->popupascbin83, XV_SHOW, TRUE, 0);

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        /* gxv_end_connections */

}
```

```
/*
 * sdvipro_stubs.c - Notify and event callback function stubs.
 * This file was generated by 'gxv' from 'sdvipro.G'.
 */

#include <stdio.h>
#include <sys/param.h>
#include <sys/types.h>
#include <xview/xview.h>
#include <xview/panel.h>
#include <xview/textsw.h>
#include <xview/notice.h>   /* inclui'do para acesso de notice_prompt() */
#include <xview/xv_xrect.h>
#include "sdvipro_ui.h"

#include "FSProc_ui.h"      /* Define o objeto FSProc_FSProcpoup            */
#include "popupxview_ui.h"  /* Define o objeto popupxview_popupxview_objects */


/*
 * Global object definitions.
 */
sdvipro_processamento_objects   *Sdvipro_processamento;

/*
 * Declara,cao de varia'veis para as Popup's a serem chamadas
 */
FSProc_FSProcpoup_objects       *FSProc_FSProcpoup;
popupxview_popupxview_objects   *Popupxview_popupxview;

#ifdef MAIN

/*
 * Instance XV_KEY_DATA key.  An instance is a set of related
 * user interface objects.  A pointer to an object's instance
 * is stored under this key in every object.  This must be a
 * global variable.
 */
Attr_attribute  INSTANCE;

main(argc, argv)
        int     argc;
        char    **argv;
{
        /*
         * Initialize XView.
         */
        xv_init(XV_INIT_ARGC_PTR_ARGV, &argc, argv, NULL);
        INSTANCE = xv_unique_key();

        /*
         * Initialize user interface components.
         * Do NOT edit the object initializations by hand.
         */
        Sdvipro_processamento = sdvipro_processamento_objects_initialize(NULL, N
ULL);

        /* Incializa o processamento da Pop_up window  FSProcpoup */
        FSProc_FSProcpoup = FSProc_FSProcpoup_objects_initialize(NULL, Sdvipro_p
rocessamento->processamento);
```



```
        /* Incializa o processamento da Popup popupxview                */
        Popupxview_popupxview = popupxview_popupxview_objects_initialize(NULL, S
dvipro_processamento->processamento );


        /*
         * Turn control over to XView.
         */
        xv_main_loop(Sdvipro_processamento->processamento);
        exit(0);
}

#endif


/*
 * Menu handler for 'aplicativo (XV)'.
 */
Menu_item
XvHItem(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        sdvipro_processamento_objects * ip = (sdvipro_processamento_objects *) x
v_get(item, XV_KEY_DATA, INSTANCE);

        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
                /* Usa system() para chamar o XV         */
                system("xv &");
                fputs("sdvipro: XvHItem: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'aplicativo (XView)'.
 */
Menu_item
XviewerHItem(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        sdvipro_processamento_objects * ip = (sdvipro_processamento_objects *) x
v_get(item, XV_KEY_DATA, INSTANCE);
```

```
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
                xv_set(Popupxview_popupxview->popupxview, XV_SHOW, TRUE, 0);

                /* fputs("sdvipro: XviewerHItem: MENU_NOTIFY\n", stderr);  ret *
/

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'aplicativo (ftptool)'.
 */
Menu_item
FtptoolHItem(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        sdvipro_processamento_objects * ip = (sdvipro_processamento_objects *) x
v_get(item, XV_KEY_DATA, INSTANCE);
        int      result;
        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
             /* Usa o pacote notice para informar sobre a ferramenta ftptool */
             result = notice_prompt(ip->processamento, NULL,
          /* NOTICE_FOCUS_XY,        event_x(event), event_y(event), */
             NOTICE_MESSAGE_STRING,  "ftptool Ø uma ferramenta de transferncia
remota de arquivos!",
             NOTICE_BUTTON_YES,      "Continuar",
             NOTICE_BUTTON_NO,       "Cancelar",
             NULL);
             if (result == NOTICE_YES)
                system("ftptool &");         /* Para chamar ftptool */

                fputs("sdvipro: FtptoolHItem: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */
```



```
                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'Usokhoros (Cantata)'.
 */
Menu_item
fcantata(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        sdvipro_processamento_objects * ip = (sdvipro_processamento_objects *) x
v_get(item, XV_KEY_DATA, INSTANCE);

        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
                /* Usa system() para chamar o cantata utilitÁrio do Khoros */
                system("cantata &");
                fputs("sdvipro: fcantata: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'Usokhoros (Xprism2)'.
 */
Menu_item
fxprism2(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        sdvipro_processamento_objects * ip = (sdvipro_processamento_objects *) x
v_get(item, XV_KEY_DATA, INSTANCE);

        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;
```

```
        case MENU_NOTIFY:
                /* Usa system() para chamar o Xprism2 utilitÆrio do Khoros */
                system("xprism2 &");

                fputs("sdvipro: fxprism2: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'Usokhoros (Xprism3)'.
 */
Menu_item
fxprism3(item, op)
        Menu_item       item;
        Menu_generate   op;
{
        sdvipro_processamento_objects * ip = (sdvipro_processamento_objects *) x
v_get(item, XV_KEY_DATA, INSTANCE);

        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
                /* Usa system() para chamar o Xprism3 utilitÆrio do Khoros */
                system("xprism3 &");

                fputs("sdvipro: fxprism3: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Menu handler for 'Usokhoros (Warpimage)'.
 */
Menu_item
fwarpimage(item, op)
        Menu_item       item;
```



```
        Menu_generate   op;
{
        sdvipro_processamento_objects * ip = (sdvipro_processamento_objects *) x
v_get(item, XV_KEY_DATA, INSTANCE);

        switch (op) {
        case MENU_DISPLAY:
                break;

        case MENU_DISPLAY_DONE:
                break;

        case MENU_NOTIFY:
                /* Usa system() para chamar o Warpimage utilitÆrio do Khoros */
                system("warpimage &");
                fputs("sdvipro: fwarpimage: MENU_NOTIFY\n", stderr);

                /* gxv_start_connections DO NOT EDIT THIS SECTION */

                /* gxv_end_connections */

                break;

        case MENU_NOTIFY_DONE:
                break;
        }
        return item;
}

/*
 * Notify callback function for 'button2'.
 */
void
fsistema(item, event)
        Panel_item      item;
        Event           *event;
{
        sdvipro_processamento_objects *ip = (sdvipro_processamento_objects *) xv
_get(item, XV_KEY_DATA, INSTANCE);
         /* Usa o pacote notice para informar sobre as funcoes do sistema  */
        int      result;
        xv_set(FSProc_FSProcpoup->FSProcpoup, XV_SHOW, TRUE, 0);
       result = notice_prompt(ip->processamento, NULL,
             NOTICE_FOCUS_XY,        event_x(event), event_y(event),
             NOTICE_MESSAGE_STRING,  "Adicionar programas em Janela, conforme  \
n forem desenvolvidos em Linguagem C. ",
             NOTICE_BUTTON_YES,      "OK",
             NULL);
         if (result == NOTICE_YES)
             fputs("sdvipro: fsistema\n", stderr);

        /* gxv_start_connections DO NOT EDIT THIS SECTION */

        /* gxv_end_connections */

}
```

Bibliografia

[1] OpenWindows Developer's Guide 1.1 User's Manual - June 1990

[2] Dan Heller, " Xview Programming Manual" by O'Reilly Associates, Inc.

[3] B.P. Lathi, Sistemas de Comunicação - 1979

[4] Michel Fortier, "VIDS-A visual Display Workstation", INRS - Télécommunications, 3, Commerce Place, Verdun, PQ - Canadá - H3E 1H6.

[5] Fortier, M., Dubois, E., "Implementation of a Programmable System for Real-Time Digital Video Processing", SMPTE Journal, October 1989.

[6] Johnston, R., Mastronardi, J., Mony, "A Digital Television Sequence Store", IEEE Tr. Comp., C26, May 1978, pp.594-600.

[7] Elio P.S.Filho, Yuzo Iano "Interfaces de Entrada e Saída para um CODEC de Vídeo", Tese de mestrado apresentada à Faculdade de Eng. Elétrica da Universidade Estadual de Campinas, Set.1989.

[8] Fortier, M., Dubois, E., and sabri, s., "Real-Time Video Simulation (RVS)", Proceedings Eletronic Imaging/88, Anahein, CA March 28-31, 1988.

[9] Theodore S. Rzeszewski, "A Technical Acessement of Advanced Television", Proceedings of the IEEE, Vol.78, Num.5, May 1990.

[10] Elio P.S.Filho, Yuzo Iano, "Interfaces A/D e D/A para codificação digital de sinal de TV PAL-M", 7º Simpósio Brasileiro de Telecomunicações, 3 a 6 de Setembro de 1989 - Florianópolis - SC;

[11] DeskSet Reference Guide - Sun Microsystems, Inc-1991.

[12] SunOs. Manual de Referência do SunOs. Man Pages.

[13] C Programmer's Guide, Sun Microsystems, Inc-1988.

[14] Afonso O. Alonso, João B.T. Yabu-uti, Normands Alens, Yuzo Iano "Frequência de Amostragem para Codificação Composta do Sinal de Vídeo PAL-M",Set 1981.

[15] Afonso O. Alonso, João B.T. Yabu-uti, Normands Alens, Yuzo Iano "Digitalização de Sinais de TV", Dez 1980.

[16] Khoros Manual Vol.I e II University of New México - May 1991.

[17] Open Look Application Style Guidelines (Manuals)

[18] Open Look Functional Specification (Manuals)

[19] Antônio M. F. Gaspar, Yuzo Iano, "Projeto de Interfaces Padrão M e Geração de Banco de Dados para um Sistema de Aquisição e Visualização de Imagens Digitalizadas." X Congresso Chileno de INGENIERIA ELECTRICA, 16 a 20 de Novembro de 1993 - Universidade Austral do Chile, Valdivia - CHILE.

[20] Antônio M. F. Gaspar, Yuzo Iano, "Uma Proposta para Implementeção de um Sistema de Aquisição e Visualização de Imagens Digitalizadas para Aplicações em PDI." X Congresso Chileno de INGENIERIA ELECTRICA, 16 a 20 de Novembro de 1993 - Universidade Austral do Chile, Valdivia - CHILE.

[21] Antônio M. F. Gaspar, Yuzo Iano, "Desenvolvimento de Interfaces em Hardware e Software para a Estrutura Básica do Sistema Digital de Visualização de Imagens - SDVI." 11º Simpósio Brasileiro de Telecomunicações, 06 a 10 de Setembro de 1993 - Universidade Federal do Rio Grande do Norte - UFRN - Natal, RN.

[22] Antônio M. F. Gaspar, Yuzo Iano, "Proposta de um Sistema Moderno para Digitalização de Imagens visando desenvolvimento tecnológicos em PDI." RPIC'93, Dezembro de 1993 - Universidad Nacional de Tucumán, FACULDAD DE CIENCIAS EXACTAS Y TECNOLOGIA - Rep. ARGENTINA.

[23] Antônio M. F. Gaspar, Yuzo Iano, "Proposta de Projecto de Interfaces padrão M para um Sistema de Aquisição e Visualização de Imagens Digitalizadas." RPIC'93, Dezembro de 1993 - Universidad Nacional de Tucumán, FACULDAD DE CIENCIAS EXACTAS Y TECNOLOGIA - Rep. ARGENTINA.

[24] A. M. A. Nascimento "Novas investigações comparativas de arquiteturas para redução da taxa de bits de sinais de tv PAL-M em 4fsc usando modelo de campo QUINCUNX". Tese de mestrado apresentada à Faculdade de Eng. Elétrica da Universidade Estadual de Campinas, Julho 1994.

[25] L.Sousa, J.Caeiro, M.Piedade, "An advanced architecture for image processing and analysis", IEEE International Symposion on Circuits and Systems, ISCAS 91, Singapure.