Universidade Federal de Campina Grande - UFCG

Implementação em Hardware de um Sistema para Extração de Objetos de um Fundo Não Homogêneo em Tempo Real.

Jozias Parente de Oliveira

Dissertação de Mestrado submetida ao Programa de Cursos de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande – Campus I, como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

Área de concentração: Processamento da Informação

Raimundo Carlos Silvério Freire, Dr.
Elmar Uwe Kurt Melcher, Dr.
Orientadores

Campina Grande. Paraíba. Brasil

O48i Oliveira Jozias Parente de
2003 Implementação em Hardware de um Sistema para Extração de Objetos de um fundo não Homogêneo em tempo Real /Jozias Parente de Oliveira
132p.: il.
Inclui bibliografia
Dissertação (Mestrado em Engenharia Elétrica) - UFCG/CCT/DEE

1. Corpo Virtual 2. Sistemas Distribuídos 3. Sistemas de Comunicação Pessoal 4. Tecnologia JINI

CDU: 621.397

IMPLEMENTAÇÃO EM HARDWARE DE UM SISTEMA PARA EXTRAÇÃO DE OBJETOS DE UM FUNDO NÃO HOMOGÊNEO EM TEMPO REAL

JOZIAS PARENTE DE OLIVEIRA

Dissertação Aprovada em 23.09.2003

ELMAR UWE KURT MELCHER, Dr., UFCG
Orientador

RAIMUNDO CARLOS SILVÉRIO FREIRE, Dr., UFCG
Orientador

JOÃO MARQUES DE CARVALHO, Ph.D.,UFCG
Componente da Banca

JOSEANA MACEDO FECHINE, D.Sc., UFCG
Componente da Banca

CAMPINA GRANDE - PB
Setembro - 2003

Dedicatória

Dedico este trabalho a minha esposa Rosiele pelos incentivos e compreensão, a minha mãe Izabel pela motivação e orações e a memória de meu pai José Parente que sempre incentivou a minha formação.

Agradecimentos

Gostaria de expressar os meus sinceros agradecimentos à Coordenação de Pós-Graduação em Engenharia Elétrica (COPELE) da UFCG pela oportunidade do Mestrado à Distância; Aos professores Raimundo Freire e Elmar Melcher por acreditarem no meu trabalho e pela viabilidade deste projeto; Aos membros da banca examinadora: João Marques de Carvalho e Joseana Macedo Fechine pelas críticas e sugestões; Aos colegas do Genius Instituto de Tecnologia: André Braga, André Printes, Claudionor Ramos, Dieter Schwanke, Marcel Bergeman e José Eduardo Vianna pela contribuição ao longo deste trabalho.

Resumo

Apresenta-se neste trabalho a definição e implementação de um sistema em *hardware* para extração de objetos de uma imagem com fundo heterogêneo, porém estático, em tempo real, com robustez a pequenas variações de iluminação global e local na imagem.

O sistema de extração de objetos foi implementado na plataforma denominada GPIP01 (*General Purpose Image Processor*), desenvolvida por André Printes, com algumas adaptações necessárias para o desenvolvimento do sistema de extração de objetos.

O sistema de extração de objetos foi submetido a testes em um ambiente real para avaliação de desempenho do método de extração implementado em *hardware*. São apresentados os resultados dos testes realizados e a avaliação do sistema quanto a sua capacidade de extração de objetos de uma imagem heterogênea, de distinção entre objetos da imagem de fundo e da imagem de primeiro plano com cores similares, e robustez a variações de iluminação e sombras na imagem de fundo.

Abstract

It is presented in this thesis the implementation and definition of a hardware system for extraction of objects from images with static and non-homogeneous backgrounds in real time, robust with respect to minor global and local brightness variations.

The object extraction system was implemented on the GPIP01 (General Purpose Image Processor) platform designed by André Printes. Little modifications on the platform were done for the implementation of the system.

The object extraction system was submitted to actual environment situations to test the performance of the extraction method implemented in hardware. The results of the tests and evaluation of the system demonstrate its capability to extract object from non-homogeneous backgrounds, to deal with differences between background objects and foregrounds objects with the same color and its tolerance to brightness variations and shadows in the background.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

Símbolo	Descrição
i	Posição (x,y) qualquer de um *pixel* em um quadro.
E_i	Valor esperado para o *pixel i* na imagem de fundo. É formado por $\mu_{i(R)}, \mu_{i(G)}, \mu_{i(B)}$
$\mu_{i(R)}, \mu_{i(G)}, \mu_{i(B)}$	Valor médio de cada uma das componentes R,G e B do *pixel i* na imagem de fundo, calculado em um número N de quadros.
I_i	Valor do *pixel i* na imagem do primeiro plano. É formado por $I_{i(R)}, I_{i(G)}, I_{i(B)}$
α_i	Distorção de Brilho.
$I_{i(R)}, I_{i(G)}, I_{i(B)}$	Cada uma das componentes R, G e B do *pixel i* na imagem do primeiro plano.
CD_i	Distorção de cor
S_i	Desvio padrão dos valores de cada um dos *pixels* de uma imagem estática de fundo. É formado por $\sigma_{i(R)}, \sigma_{i(G)}, \sigma_{i(B)}$
$\sigma_{i(R)}, \sigma_{i(G)}, \sigma_{i(B)}$	Desvio padrão de cada uma das componentes R, G e B de um *pixel i* na imagem de fundo, calculado em um número N de quadros.
a_i	Variação da distorção de brilho, fator de normalização.
b_i	Variação da distorção de cor, fator de normalização.
$\hat{CD}_i$	Distorção de cor normalizada
$\hat{\alpha}_i$	Distorção de brilho normalizada
τ_{CD}	Limiar de decisão para $\hat{CD}_i$.
$\tau_{\alpha 1}$	Limiar superior para $\hat{\alpha}_i$.
$\tau_{\alpha 2}$	Limiar inferior para $\hat{\alpha}_i$.
τ_{alo}	Limite inferior para o valor da distorção de brilho normalizada.
$P_{i(C)}$	Valor de um *pixel* i sendo C cada uma das componentes de cor R, G e B.
$M_{(i)}$	Máscara para classificação dos *pixels*

1 Introdução

1.1 Motivação

O interesse em métodos de processamento de imagens digitais decorre de duas áreas principais de aplicação: melhoria de informação visual para a interpretação humana e o processamento de dados de cenas para percepção automática por máquinas [1]. A melhoria do contraste e codificação de cores de imagens em raios X ou outras imagens biomédicas, o estudo de padrões de poluição em imagens aéreas ou de satélites e a restauração de figuras fotografadas borradas são exemplos de melhoria da informação nas áreas de medicina, geografia e arqueologia, respectivamente [1]. A outra área de aplicação de técnicas de processamento de imagens digitais se concentra em procedimentos para extrair de uma imagem informações úteis para posterior processamento computacional. Um problema típico desta área é a extração de objetos de um fundo conhecido. O processamento fundamental para extração de objetos é a segmentação. A segmentação consiste em dividir uma imagem em segmentos que são regiões de interesse, representando objetos significativos para o usuário. Uma região de interesse refere-se a um grupo de *pixels*[1] na imagem que compartilham uma característica comum. Esta característica pode ser brilho, cor, textura, movimento ou outras, ou mesmo uma combinação de várias características [2]. As informações obtidas de cada segmento individual podem ser usadas em processamentos subseqüentes para classificação destes segmentos.

No decorrer dos últimos anos, diversos métodos têm sido desenvolvidos para extração de objetos de um fundo homogêneo e conhecido em ambientes controlados [3], extração de objetos de fundo não homogêneo [4], extração de objetos com atualização automática de fundo [5] e extração de objetos em fundos desconhecidos [6]. Basicamente os métodos de extração de objetos consistem em distinguir um objeto, pertencente a uma

[1] *Pixel* – Derivado da expressão em inglês *picture element* (elemento de imagem).

imagem de primeiro plano, de uma imagem de fundo previamente analisada. Umas das aplicações dos métodos de extração de objetos ocorre em estúdios de produção de vídeo em que o método denominado *Chroma Key* é muito utilizado. O objetivo deste método é extrair uma imagem de primeiro plano de uma imagem de fundo uniforme e homogêneo conhecido e inserir os *pixels* do objeto em uma segunda imagem [3]. Uma outra aplicação dos métodos de extração de objetos ocorre na área de segurança para detecção de intrusos.

A aplicação do método implementado neste trabalho visa o mercado de eletrônica de consumo com aplicações, por exemplo, na área de segurança para detecção de intrusos ou de entretenimento inserido em jogos eletrônicos. Desta maneira, os requisitos essenciais são os seguintes:

- **Baixo custo**: No mercado de eletrônica de consumo um produto é competitivo se oferecer ao consumidor um custo relativamente baixo e uma boa qualidade de desempenho. Em função do alto volume de produção alguns centavos no custo de fabricação são significativos no preço do produto final. Para atender ao requisito de baixo custo implementou-se um método de extração de objetos que utiliza operações de baixa complexidade para implementação em *hardware*.
- **Simplicidade de uso**: O procedimento para operação do produto deve ser o mais simples possível afim de que todos os usuários possam utilizar facilmente todas as funções do produto. Para atender ao requisito de simplicidade de uso o sistema implementado neste trabalho requer apenas que o usuário pressione uma chave para iniciar as etapas de aprendizagem da imagem de fundo e segmentação do objeto.
- **Robustez a sombras e a pequenas variações de iluminação do ambiente**: As sombras podem ser facilmente geradas por objetos em movimento na imagem de primeiro plano e a iluminação do ambiente pode variar em função da simples abertura ou do fechamento de uma janela no ambiente em que o sistema esteja inserido. O método implementado neste trabalho se propõe a minimizar os erros de detecção gerados por sombras e variações de iluminação na imagem.

1.2 Objetivos

Nesta dissertação continua-se o trabalho de Printes [7], que apresentou uma plataforma para o desenvolvimento de um sistema em *hardware* capaz de capturar uma imagem, extrair o objeto principal de um fundo heterogêneo qualquer, porém estático, e inseri-lo em uma segunda imagem de fundo diferente, oriunda de outra fonte de vídeo em tempo real. Printes [7] desenvolveu uma plataforma genérica para sistemas de processamento de imagens, adaptou um método estatístico para extração de objetos de um fundo heterogêneo, validou o método adaptado no MatLab [8] e apresentou os diagramas em blocos e as máquinas de estados teóricas para implementação em *hardware* deste método.

O propósito desta dissertação é implementar em hardware o método de extração de objetos de um fundo não homogêneo, utilizando a plataforma GPIP-01 (*General Purpose Image Processor*) [7] desenvolvida para este propósito, concluindo a etapa de extração do objeto do sistema, restando ainda as etapas de inserção deste objeto extraído em uma segunda imagem e o desenvolvimento de uma interface USB para utilização de uma câmera de baixo custo, que estão fora do escopo deste trabalho e podem ser desenvolvidos em trabalhos futuros.

A fim de implementar-se a etapa de extração do objeto com baixo custo, algumas otimizações serão realizadas nas máquinas de estados propostas em [7], com o intuito de compartilhar-se alguns blocos comuns, reduzindo a quantidade de elementos lógicos utilizados na implementação do sistema.

Uma análise de desempenho do sistema implementado em hardware será efetuada visando avaliar a robustez do sistema em tempo real quanto a pequenas variações de iluminação no ambiente.

1.3 Organização do Documento

Esse documento é dividido em 6 capítulos, além das referências bibliográficas.

No Capítulo 2 são apresentados alguns fenômenos que ocorrem em ambientes não controlados e que devem ser considerados na escolha do método de extração de objetos. Em seguida, apresentamos alguns métodos pesquisados e uma comparação qualitativa de desempenho destes métodos. Finalmente, estabelece-se os requisitos do sistema implementado neste trabalho e os motivos para escolha da solução proposta.

No Capítulo 3 são apresentados os fundamentos do método proposto para extração de objetos e as adaptações necessárias para implementação do método proposto em hardware.

No Capítulo 4 é apresentada a plataforma de desenvolvimento do sistema. Em seguida, faz-se uma análise dos requisitos de hardware e a descrição dos blocos do sistema. Finalmente, apresentam-se os procedimentos para validação comportamental e estrutural dos blocos do sistema.

No Capítulo 5 são apresentados os resultados atingidos com a implementação do sistema em hardware em tempo real.

No Capítulo 6 são apresentadas as conclusões e são descritas as sugestões para trabalhos futuros.

2 Métodos para Extração de Objetos

A maioria dos sistemas que detectam e reconhecem um objeto em uma imagem necessita realizar a extração do objeto através da segmentação da imagem e identificação dos *pixel*s de imagem pertencentes ao fundo e a imagem do primeiro plano. O algoritmo utilizado para este processamento deverá considerar alguns fenômenos que comumente ocorrem nos ambientes não controlados em que o sistema esteja inserido. Os problemas mais comuns são [9]:

- **Mudança de iluminação:** O fundo da imagem sofre modificações com a iluminação e tais mudanças podem ser confundidas com um objeto da imagem principal.

- **Sombras e reflexões:** O movimento de objetos na imagem principal pode gerar mudanças em parte da iluminação local da imagem de fundo.

- **Mudanças de objetos da imagem de fundo:** Quando objetos são retirados ou acrescentados à cena ou quando objetos considerados da imagem de fundo são modificados, é necessário atualizar o modelo da imagem de fundo caso a mudança persista por um longo período.

- **Camuflagem:** A similaridade entre a aparência dos objetos da imagem principal e da imagem de fundo dificulta a distinção entre elas.

- **Imagem de fundo não-estática:** As mudanças na imagem de fundo como, por exemplo, um ventilador em funcionamento em frente a uma parede não é fácil de se modelar com um simples modelo estático do *pixel*.

- **Áreas "congestionadas":** Quando a imagem de fundo é freqüentemente obstruída por diferentes objetos da imagem principal, pode ser difícil determinar a imagem de fundo e extrair o objeto desta imagem.

- **Identificação da imagem de fundo:** Em alguns ambientes o aprendizado da imagem de fundo sem a presença de outro objeto estranho não é possível [10].

- **Imagem principal estática:** Um objeto da imagem principal que esteja estático não pode ser distinguido de um objeto da imagem de fundo que modificou e em seguida tornou-se estático [9].

Não há sistema perfeito [9]. Alguns dos problemas citados anteriormente são contornados com técnicas computacionais complexas e de alto custo. No caso de sistemas para processamento em tempo real e de baixo custo são utilizadas técnicas que realizam o processamento num curto período de tempo e que estabelecem algumas restrições quanto aos problemas citados anteriormente.

2.1 Contexto

Muitos métodos para extração de objetos foram propostos nas últimas décadas. O método mais simples, freqüentemente encontrado em sistemas experimentais de computação, utiliza a cor ou intensidade de brilho como parâmetros de entrada e um modelo de imagem de fundo que representa os valores das características esperadas de cada *pixel* na imagem de fundo. As estatísticas da imagem de fundo são freqüentemente modeladas para cada *pixel* e armazenadas em um modelo de referência.

O modelo de referência é construído durante uma fase de aprendizado enquanto a cena, imagem de fundo, está sem nenhum objeto da imagem do primeiro plano. Na fase de segmentação, a imagem do primeiro plano é detectada *pixel* a *pixel* sempre que difere significativamente da distribuição dos valores esperados para a imagem de fundo contidos no modelo de referência .

Muitos métodos tentam resolver alguns, mas nunca todos os problemas apresentados anteriormente, melhorando um ou mais aspectos do método de segmentação. Por exemplo, atualizando a imagem de fundo constantemente, o sistema pode adaptar-se a mudanças graduais de iluminação, sem a necessidade de uma cena sem objetos para inicialização e pode incorporar alterações de objetos no modelo da imagem de fundo. Por outro lado, a inicialização de tais sistemas requer um longo período a fim de que o algoritmo consiga detectar apenas a imagem de fundo sem os objetos em movimento.

Todos estes modelos ainda apresentam dificuldades para solucionar os problemas que ocorrem no ambiente em que o sistema está inserido, conforme mencionado anteriormente.

A seguir são apresentados alguns métodos propostos para extração de objetos e suas principais características.

2.1.1 Diferença entre Quadros Adjacentes

Devido a sua simplicidade, o método denominado Diferença entre Quadros Adjacentes é muito utilizado para detecção de mudanças em imagens. Aplicações comuns incluem detecção de objetos [11], sistema de vigilância [12, 13], sistema de vigilância de veículos [14, 15 , 16], análise de imagens de satélites [17] para medir erosão da terra e desflorestamento e análise de imagens médicas para medição de distribuição celular [18], entre outras.

Na Figura 2-1 mostra-se um diagrama em blocos representativo do método Diferença entre Quadros Adjacentes. Cada imagem corrente (K+1) é subtraída da imagem anterior (K) consecutivamente, resultando em uma terceira imagem que corresponde à diferença entre a imagem atual e a anterior. Esta imagem resultante é comparada com um limiar de referência fixo predeterminado, permitindo a identificação de locais de mudança na cena. Diferenças absolutas maiores que o limiar são classificadas como imagem do primeiro plano.

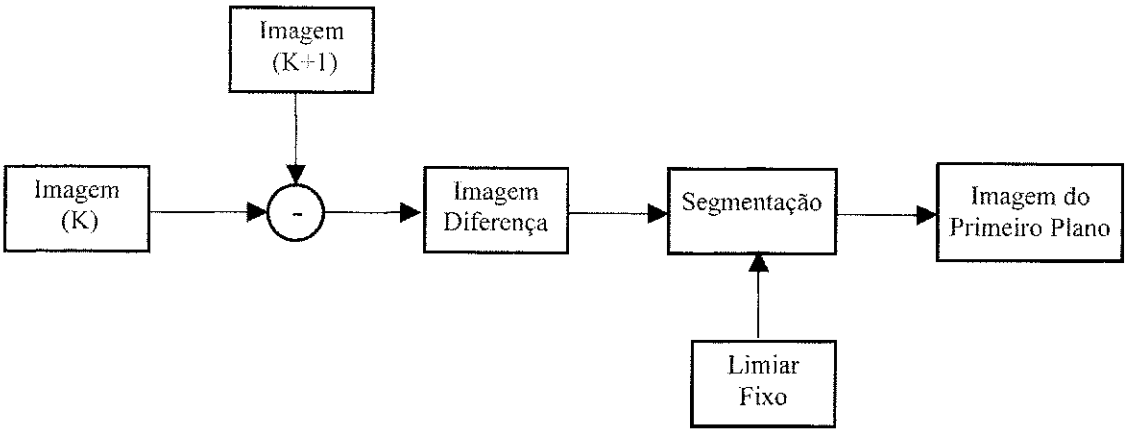


Figura 2-1: Diagrama em blocos do método Diferença entre Quadros Adjacentes.

Os valores dos limiares são extremamente críticos, visto que, valores baixos ocasionam a detecção de sinais espúrios na imagem, enquanto que valores altos suprimem mudanças significativas na imagem. O valor adequado do limiar de comparação depende da cena, das possíveis condições de variação do nível de sinal da câmera, bem como das condições de iluminação que constantemente podem sofrer modificações [19]. Desta forma, para assegurar a robustez do sistema, os limiares devem ser baseados no conteúdo da imagem.

Rosin [19] apresenta alguns métodos para determinação dos limiares adequados para detecção de mudanças na imagem, baseados em estimativas do comportamento do ruído na imagem, obtendo-se uma curva de distribuição da qual são estabelecidos os limiares de comparação. Por se tratar de uma estimativa, estes modelos não oferecem muita robustez ao sistema. Nas seções subseqüentes são apresentados métodos que utilizam características da imagem, tais como: imagem média, desvio médio, variância entre outras, para determinação dos limiares.

A detecção de mudanças na imagem por diferenciação de quadros adjacentes apresenta, entre outras, a desvantagem de omitir objetos estacionários na cena [20]. Uma alternativa para solucionar esta deficiência seria a diferenciação da seqüência de imagens em relação a uma imagem de referência, proposta nos métodos descritos nas seções 2.1.2 a 2.1.6.

2.1.2 Média e Limiar

Na Figura 2-2 mostra-se um diagrama em blocos representativo do método denominado Média e Limiar. Este método é subdividido em 3 partes: primeiramente a geração de uma referência adequada ou imagem de fundo; segundo, a subtração da imagem corrente da imagem de referência e terceiro, a seleção e aplicação do limiar adequado [20].

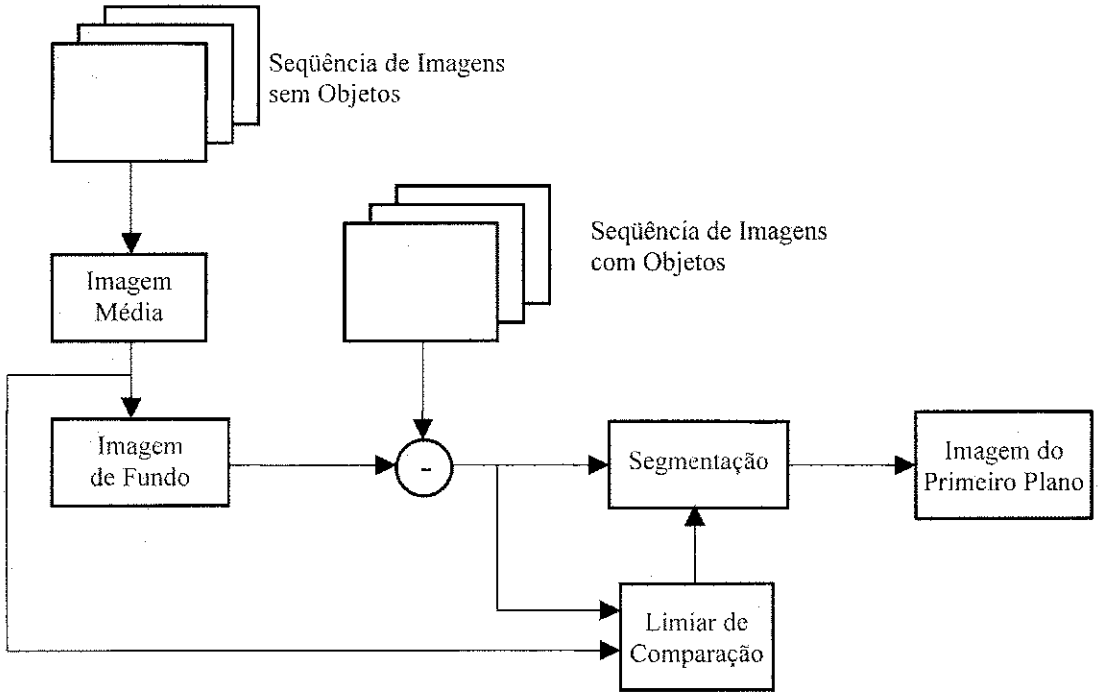


Figura 2-2: Diagrama em blocos do método Média e Limiar.

A imagem de referência é adquirida durante um período de relativa inatividade dentro da cena. Durante esta fase de aprendizado, são adquiridos alguns quadros da imagem de fundo para determinação de uma imagem média que será armazenada no sistema. Finalizada a fase de aprendizado, as novas imagens são subtraídas da imagem de referência, gerando a diferença *pixel* a *pixel* entre as imagens. A partir da imagem diferença e da imagem média são determinados os limiares de comparação de cada *pixel*. A extração de objetos da imagem é feita comparando-se as novas imagens *pixel* a *pixel* com os limiares.

Rosin e Ellis [20] determinam os limiares automaticamente através do cálculo da mediana e do desvio médio entre a imagem de referência e as novas imagens.

A determinação dos limiares baseada na imagem média de referência aumenta a robustez do sistema, porém ainda estará sujeito a erros de detecção em função de sombras e variações de iluminação na cena. Métodos mais sofisticados atualizam constantemente a imagem de referência a fim de adaptar-se as mudanças de iluminação da imagem [5,10]. Outros são capazes de detectar sombras na imagem [4], tornando o sistema mais robusto.

2.1.3 Correlação Normalizada entre Blocos

Na Figura 2-3 mostra-se um diagrama em blocos representativo do método denominado Correlação Normalizada entre Blocos. Inicialmente há uma fase de aprendizado em que são adquiridos alguns quadros para obtenção da média e do desvio em relação à média. Cada imagem na fase de aprendizado é dividida em blocos e cada bloco tem a sua média e o desvio médio dos *pixels* agrupados no bloco.

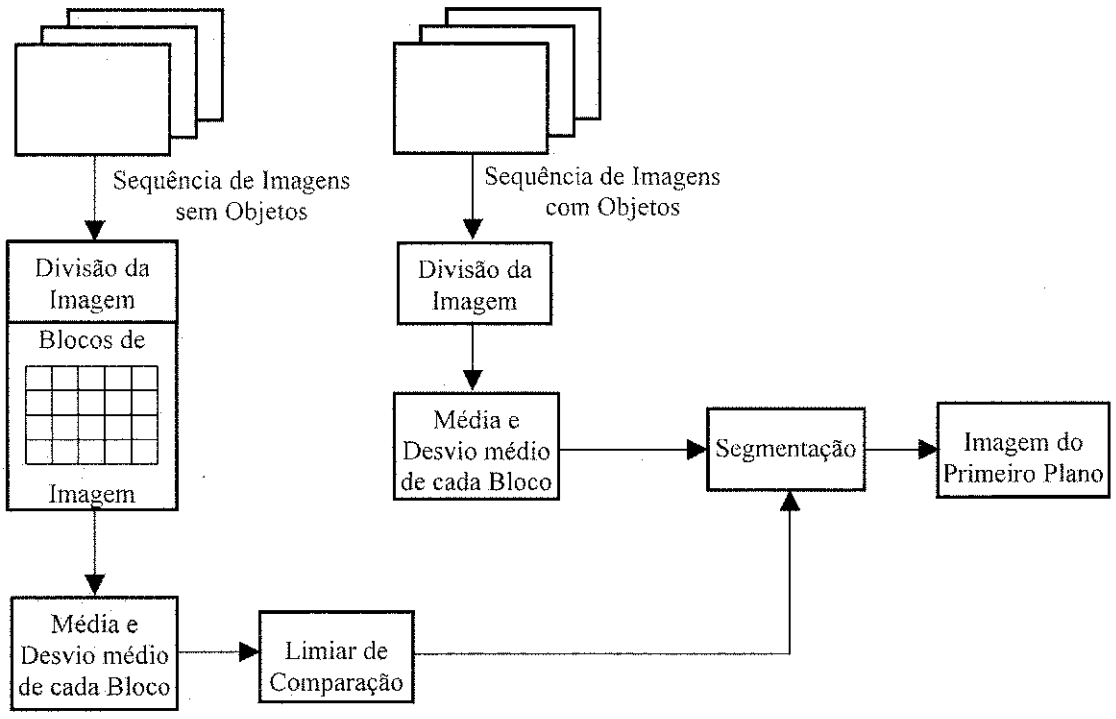


Figura 2-3: Diagrama em Blocos do método Correlação Normalizada entre Blocos.

A segmentação é realizada dividindo-se as novas imagens em blocos similares aos da fase de aprendizado e comparando-os com os blocos de referência. O bloco cujo valor

médio estiver com um desvio maior que o esperado é classificado como imagem do primeiro plano [21,22].

Apesar deste método buscar uma maior exatidão na segmentação dividindo a imagem em blocos menores, os limiares de decisão estão baseados apenas na média e no desvio em relação à média que ainda estão sujeitos a erros de detecção em função de variações de iluminação e da própria câmera. Além disso, há uma perda significativa de resolução na imagem segmentada [10].

2.1.4 Derivada Temporal

Na Figura 2-4 mostra-se um diagrama em blocos representativo do método denominado Derivada Temporal. Este método é utilizado no sistema para detecção e identificação de pessoas (W4) [23]. A segmentação utiliza também um modelo de referência da imagem de fundo adquirido em uma fase de aprendizado. Durante esta fase, cada *pixel* é representado por três parâmetros: o valor máximo, mínimo e a diferença máxima entre quadros consecutivos. A classificação do *pixel* é realizada comparando-se as novas imagens com o modelo de referência. Os *pixels* cujas intensidades mínima ou máxima forem superiores a máxima diferença esperada serão considerados pertencentes à imagem principal.

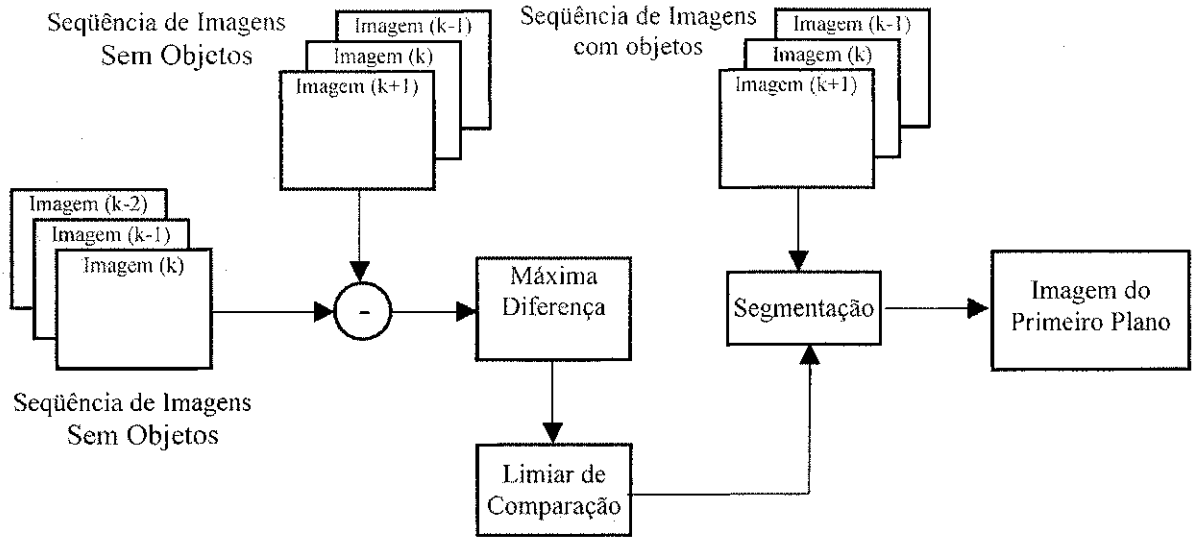


Figura 2-4: Diagrama em blocos do método Derivada Temporal.

A determinação dos limiares de comparação baseada apenas em valores máximos e mínimos também estará sujeito a erros de detecção, visto que estes valores não constituem uma medida de dispersão muito precisa, pois não levam em consideração os valores intermediários dos *pixels* e, pequenas variações na cena ou flutuações na câmera, podem induzir erros de detecção no sistema.

2.1.5 Chroma Key[2]

O método denominado *Chroma Key* é muito utilizado em estúdios de produção de vídeo. O objetivo deste método é extrair uma imagem de primeiro plano de um fundo uniforme e homogêneo conhecido e inserir os *pixels* do objeto em uma segunda imagem [3]. A técnica se destina a extração de objetos de um fundo homogêneo e conhecido em ambientes com condições controladas. Esse método apresenta um bom desempenho no processo de extração, porém não se destina a extração de objetos de um fundo heterogêneo.

2.1.6 Método Estatístico de Horprasert

Horpraset et al. [4] estabeleceu um método estatístico para extração de objetos de fundos não homogêneos e desconhecidos. O método apresenta robustez às sombras e variações originadas pelos sensores da câmera.

A primeira fase deste método consiste na aquisição de uma imagem de referência e do cálculo de parâmetros estáticos que expressam as características da imagem de fundo, tais como: variação de brilho e variação de cor. Na primeira etapa da fase de aquisição da imagem de fundo são adquiridos alguns quadros utilizados para obtenção de uma imagem média. Em seguida, uma nova seqüência de imagens é adquirida para a determinação do desvio padrão dos *pixels* da imagem. Este parâmetro faz uma estimativa das variações de

[2] *Chroma Key*: Termo em inglês que significa chave de cor. Nesse texto usa-se o termo em inglês por já ser de uso corrente.

cada *pixel* em função das variações geradas pelos sensores da câmera. Na terceira etapa, novas imagens são adquiridas e processadas para determinação das distorções de brilho e de cor de cada *pixel* na imagem, isto é, a "diferença" de brilho e cor dos *pixel*s de referência em relação aos novos *pixel*s adquiridos. Em seguida, é determinada uma média de distribuição das distorções de brilho e cromaticidade que expressa a variação destas distorções. A partir destes valores são gerados histogramas para distorção de brilho e cor e automaticamente são estabelecidos os limiares de comparação. Desta forma, é finalizada a etapa de aprendizado da imagem de fundo.

Para extração de objetos da imagem de fundo são calculadas as distorções de brilho e cor para as novas imagens e comparadas com os limiares determinados na fase de aprendizado.

Além dos métodos supracitados, há outros mais complexos tais como: utilização de vários modelos gaussianos para segmentação de imagens [9, 24, 25, 26], modelo estático geométrico [6] que utiliza duas câmeras e um mapa de disparidade para extração de objetos de um fundo e o *Wallflower* [10] que combina técnicas de análise de *pixel*, predição linear e atualização da imagem de fundo.

2.2 Comparação dos Métodos Pesquisados

Neste tópico são apresentados os resultados do estudo comparativo entre 10 métodos utilizados para subtração de imagem realizado por Toyama [10] em diversas situações que geram os problemas citados no início do capítulo. Em seguida são feitas algumas considerações acerca do método proposto neste trabalho. Convém ressaltar que os métodos Predição Linear, *Wallflower, Eigenbackground, Bayesian Decision,* Mistura de Modelos Gaussianos e o Média e Covariância não foram analisados neste trabalho.

2.2.1 Aspectos Gerais dos Métodos Pesquisados

O estudo comparativo de Toyama [10] foi realizado com seqüências de imagens oriundas de uma câmera com quadros de 160 x 120 *pixels*, amostrados a 4Hz. Os métodos foram avaliados em diversos cenários controlados com o propósito de avaliar o desempenho do método de extração de objetos na solução dos seguintes problemas: mudança de objetos na imagem de fundo, variação gradual da iluminação, variação abrupta da iluminação, mudança constante de objetos na imagem de fundo, camuflagem, inicialização do sistema com objetos na imagem de fundo e mudança de objetos com cor homogênea na imagem de fundo.

Na Figura 2-5 são apresentados os resultados visuais do desempenho dos algoritmos em cada cena e os resultados idealmente esperados. A seguir são comentados os resultados obtidos com os métodos de extração de objetos em cada cena.

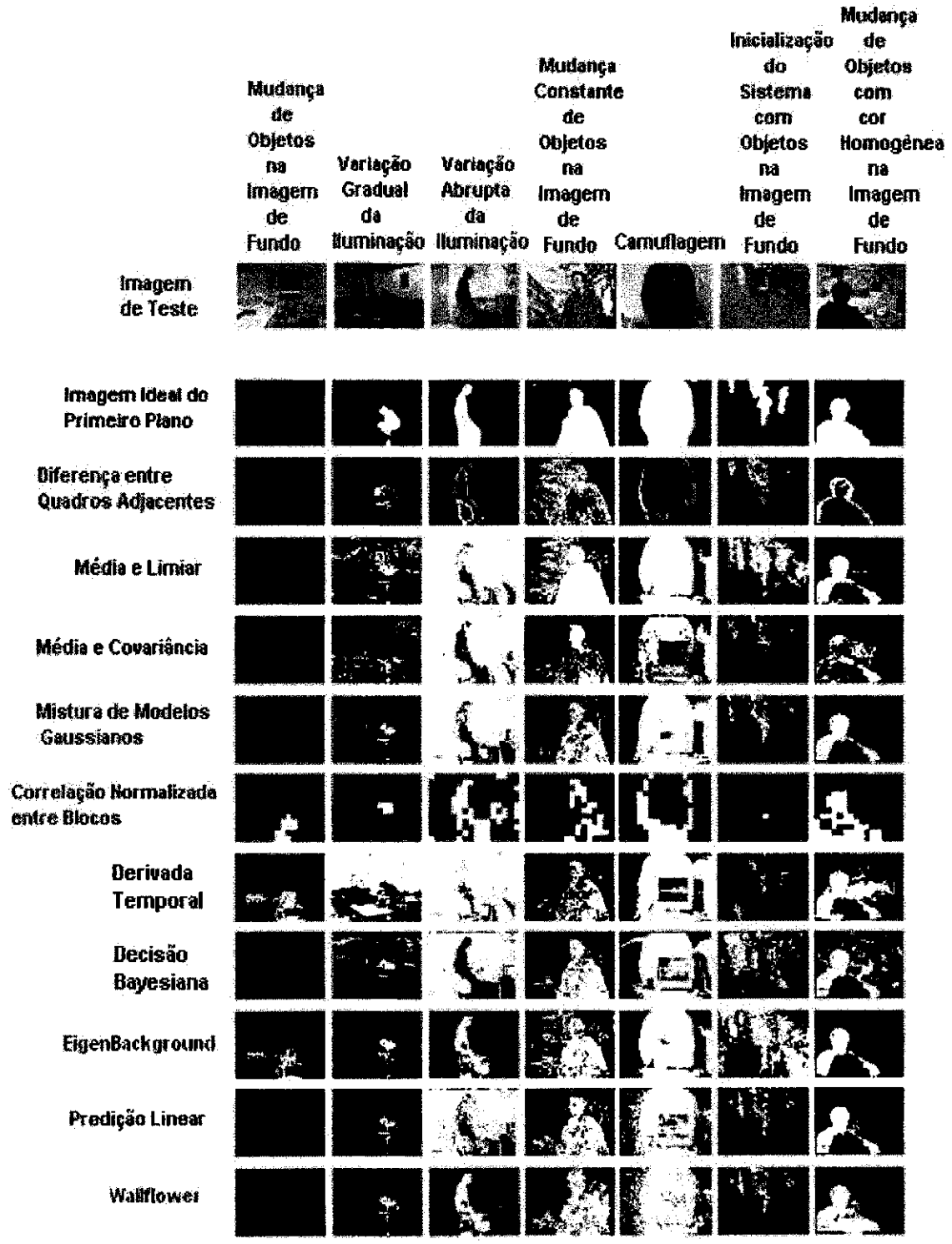


Figura 2-5: Resultados visuais dos testes realizados com os métodos em diversas situações [10].

- **Mudança de Objetos na Imagem de Fundo**: Uma pessoa entra em uma sala, faz um telefonema e modifica a posição inicial do telefone e da cadeira, simulando a mudança de objetos na imagem de fundo. Após esta modificação, aguardou-se 50 quadros para possibilitar a adaptação dos algoritmos a esta nova cena. Idealmente, a cadeira e o telefone não deveriam ser considerados como objetos do primeiro plano e, por conseguinte, a imagem deveria ser uma tela preta. Os métodos Correlação Normalizada entre Blocos, Derivada Temporal e *Eigenbackground* classificaram os *pixel*s destes objetos como pertencentes à imagem de primeiro plano. Os valores máximos, mínimos e a média dos *pixel*s na região em que os objetos foram modificados, sofreram alterações e com isso a segmentação da imagem baseada nestes parâmetros fica prejudicada.

- **Variação Gradual da Iluminação**: Uma sala escura é gradualmente iluminada. Uma pessoa entra na sala e senta-se no sofá. O intuito desta cena é modificar a imagem de fundo com variações graduais da iluminação. Idealmente, a imagem do primeiro plano deveria ser somente a pessoa sentada no sofá. Nenhum dos algoritmos identificou perfeitamente a pessoa sentada no sofá como imagem do primeiro plano. A variação de iluminação ocasionou modificações nos valores dos *pixel*s da imagem de fundo adquiridos na etapa de aprendizado e com isso os limiares deveriam ser re-calculados. O pior desempenho foi do método Derivada Temporal, visto que os valores mínimos, máximos de cada *pixel* em toda a imagem modificaram-se drasticamente com a mudança da iluminação, resultando na detecção errônea de praticamente todos os objetos da imagem de fundo como sendo da imagem do primeiro plano.

- **Variação Abrupta da Iluminação**: Primeiramente, há uma fase de treinamento em que a sala é iluminada e escurecida. Na fase de testes a sala começa com a iluminação desligada. Após alguns minutos, uma pessoa entra na sala, liga a iluminação e modifica uma cadeira do lugar. Logo em seguida, o processamento do algoritmo é interrompido propositalmente afim de evitar as adaptações do sistema. Neste caso, além da pessoa, a cadeira também é considerada como imagem do primeiro plano. O objetivo é simular modificações da imagem de fundo em função de variações abruptas da iluminação da cena. O método Diferença entre Quadros

Adjacentes classificou praticamente todos os *pixels* da imagem de fundo como pertencentes ao objeto da imagem de primeiro plano.

- **Mudança Constante de Objetos na Imagem de Fundo**: Uma árvore pertencente à imagem de fundo está com os galhos balançando e uma pessoa começa a andar em frente da árvore. Os galhos balançando simulam uma cena em que os *pixel*s da imagem de fundo não são estáticos e estão "interagindo" com os *pixel*s da imagem do primeiro plano. Idealmente, apenas a pessoa deveria ser detectada como imagem do primeiro plano. O método que apresentou o maior número de falsos positivos e negativos foi o denominado Diferença entre Quadros Adjacentes. De fato, como os galhos estão balançando, haverá diferenças nestas regiões entre quadros subseqüentes e com isso os galhos da árvore são detectados erroneamente como pertencentes à imagem do primeiro plano. Entretanto, o pior resultado visual foi apresentado pelo método Correlação Normalizada entre Blocos.

- **Camuflagem**: A imagem de fundo é constituída por um monitor, com barras de interferência "rolando" na tela, sobre uma mesa. Uma pessoa entra na sala e posiciona-se em frente ao monitor. Neste caso, o objetivo é testar a capacidade de distinguir características similares entre a imagem de fundo e a imagem principal. Idealmente, a imagem principal é somente a pessoa. O método que apresentou o melhor resultado foi o Média e Limiar.

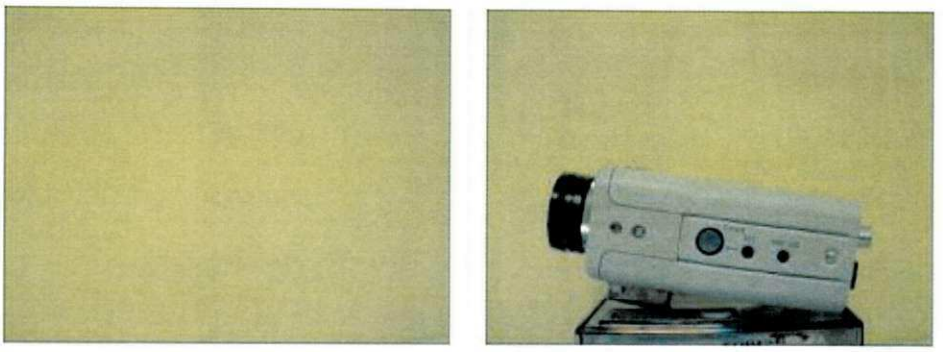
- **Inicialização do Sistema com Objetos na Imagem de Fundo**: Uma seqüência de imagens é retirada de uma lanchonete em que constantemente há pessoas na cena, simulando um ambiente no qual não há um período sem objetos na cena para realização da fase de treinamento, isto é, aprendizado da imagem de fundo. Neste caso, como a maioria dos algoritmos necessita de uma etapa de aquisição da imagem de fundo sem objetos na cena, as pessoas serão adquiridas inicialmente como imagem de fundo. No entanto, com o decorrer do teste, estas pessoas estarão em lugares diferentes em função do movimento e, desta maneira, o algoritmo identificará estas pessoas como imagem de fundo e do primeiro plano.

- **Mudança de Objetos com Cor Homogênea na Imagem de Fundo**: Uma pessoa está sentada na cadeira e com as costas voltadas para a câmera. Lentamente, ela começa a movimentar-se. A sua camisa é uniformemente colorida. Idealmente, estas modificações em função dos movimentos não deveriam ser detectadas.

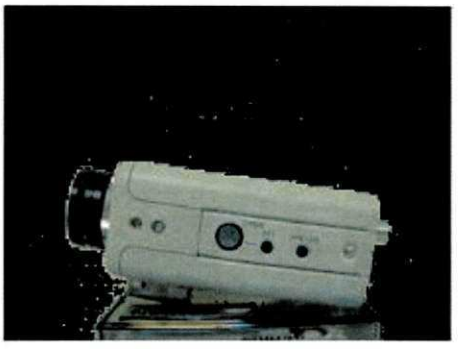
2.2.2 Resultados do Método de Horprasert

Na Figura 2-6 de (a) a (c) mostra-se o resultado em Matlab do método estatístico de Horprasert [4] adaptado por Printes [7] para uma imagem de baixa complexidade, isto é, fundo liso e monocromático, sem objetos escuros no fundo, com a cor do objeto bastante diferente da cor de fundo e praticamente sem sombras no fundo.

O processo utilizado para extração do objeto foi o seguinte: As imagens de fundo foram capturadas sem o objeto a ser extraído. Com estas imagens foram calculados os parâmetros do algoritmo, quais sejam: imagem média, desvio padrão, distorção de brilho, distorção de cor, variação da distorção de brilho, variação da distorção de cor, histogramas da distorção de brilho e cor e limiares de comparação. Na Figura 2-6a é apresentada a média de 64 quadros adquiridos da imagem de fundo. Após esta fase de aprendizado, foi inserido um objeto (Figura 2-6b) para processamento da extração. Na Figura 2-6c é apresentado o resultado obtido do processo de extração. Pode-se notar neste resultado que há *pixels* da imagem de fundo que foram identificados como pertencentes ao objeto (falsos positivos), sobretudo nas bordas do objeto em primeiro plano.



a) b)



c)

Figura 2-6: Resultado de simulação do método de Horprasert com imagem de baixa complexidade

Printes [7] também realizou testes do método de Horprasert [4] utilizando uma imagem com fundo complexo cujas características são as seguintes: fundo texturizado, objetos de fundo escuros, objetos de fundo com cor e textura similares ao objeto e objeto de fundo com variação de sombra. Na Figura 2-7 de (a) a (c) são apresentados os resultados obtidos para essa imagem de fundo. O processo utilizado para obtenção dos resultados foi o mesmo utilizado para imagem de baixa complexidade. Na Figura 2-7c é mostrado o objeto extraído da imagem de fundo. Conforme os resultados, há muitos pontos da imagem de fundo sendo classificados como pertencentes à imagem do primeiro plano e, em menor quantidade há pontos do objeto que estão suprimidos. Considerando a característica da imagem de fundo os resultados estão razoáveis, no entanto, dependendo da aplicação devem ser melhorados.

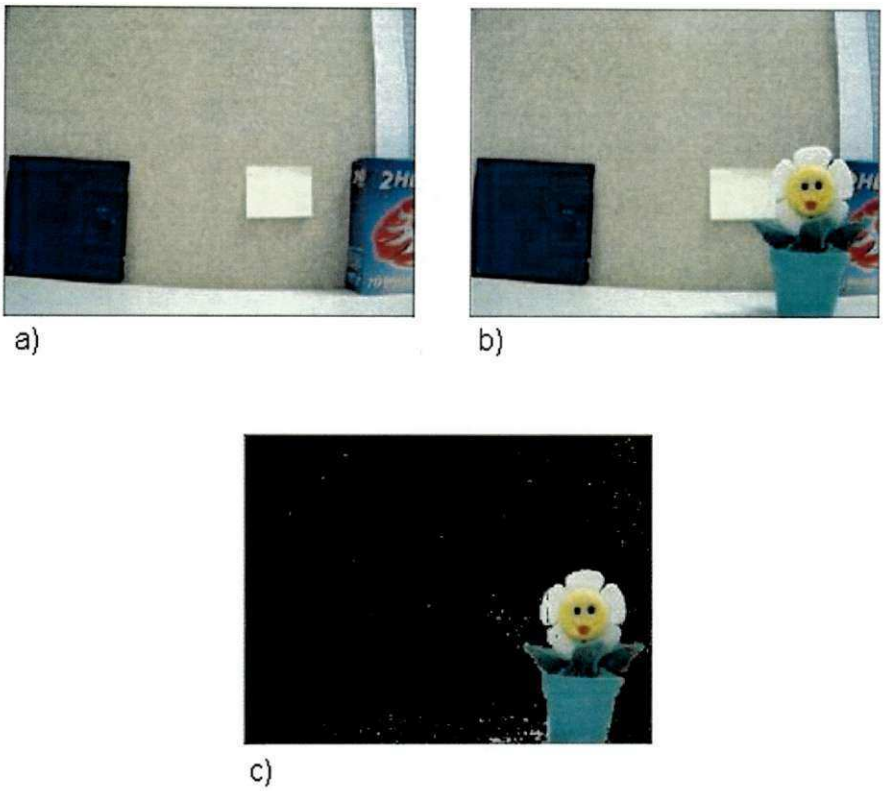


Figura 2-7: Resultado de simulação do método de Horprasert com imagem de fundo complexa

2.2.3 Conclusões

Mediante os resultados apresentados pelo estudo comparativo de Toyama [10] e pelas simulações de Printes [7], não há um algoritmo capaz de resolver todos os problemas simulados em cada cena. Portanto, a solução é estabelecer algumas limitações tendo em vista o custo e a aplicação do sistema.

2.3 Considerações Iniciais do Método Proposto

Em função dos resultados apresentados no tópico anterior e visando, sobretudo, a implementação com baixo custo em *hardware* do sistema para extração de objetos, foram estabelecidos os seguintes requisitos para o método implementado em *hardware* neste trabalho:

- O sistema utilizará uma única câmera fixa;
- A imagem de fundo será estática, porém não necessariamente homogênea;
- A fase de aprendizado, isto é, a aquisição da imagem de fundo deverá ser efetuada sem a presença de objetos que posteriormente modifiquem a sua posição inicial;
- O sistema será robusto à sombra e pequenas variações de iluminação na cena. Se houver grandes variações na iluminação da cena, o sistema deverá ser reinicializado.

Embasado nos resultados qualitativos apresentados pelo estudo comparativo de Toyama [10] e do resultado em MatLab para o método estatístico de Horpraset [4] e tendo em vista as características dos algoritmos pesquisados bem como as restrições estabelecidas para o método a ser utilizado no sistema proposto, inferiu-se o seguinte:

- Métodos que utilizam duas ou mais câmeras [6] não serão utilizados no sistema implementado neste trabalho devido à complexidade e ao alto custo de implementação;

- Métodos cuja inicialização não necessita de um período de inatividade na cena, ou seja, que constantemente atualizam a imagem de fundo [10], destinam-se a aplicações em ambientes na qual a inicialização da imagem de fundo não pode ser "controlada", tais como: sistema de vigilância de pessoas, sistema de vigilância de veículos, entre outros. No sistema proposto, a inicialização poderá ser facilmente controlada pelo usuário. Ademais, os métodos que atualizam a imagem de fundo requerem um esforço computacional considerável para esta operação e no sistema

proposto estamos visando métodos com baixo esforço computacional para implementação em *hardware*. Eliminando-se os métodos que utilizam duas ou mais câmeras e aqueles que atualizam a imagem de fundo por não satisfazerem os requisitos estabelecidos para o sistema implementado neste trabalho, restam os seguintes métodos: Diferença entre Quadros Adjacentes, Média e Limiar, Correlação Normalizada entre Blocos, Derivada Temporal e o Método Estatístico de Horprasert.

- Apesar do baixíssimo grau de complexidade de processamento, métodos que efetuam a diferença entre imagens consecutivas e um único nível de limiar para todos os *pixels* do quadro, estão sujeitos a erros de detecção e, por conseguinte, a robustez é muito baixa. Desta forma, não é um método adequado para o sistema proposto. Eliminando-se o método denominado Diferença entre Quadros Adjacentes restam apenas os seguintes métodos: Média e Limiar, Correlação Normalizada entre Blocos, Derivada Temporal e o Método Estatístico de Horprasert.

- Métodos que utilizam o valor médio dos *pixels* da imagem de fundo para determinação dos limiares de comparação [20], apesar da baixa complexidade de processamento, ainda apresentam erros de detecção de objetos quando há pequenas variações de iluminação na cena e por isso, não são adequados para o sistema proposto. Eliminando-se o método denominado Média e Limiar restam apenas os seguintes métodos: Correlação Normalizada entre Blocos, Derivada Temporal e o Método Estatístico de Horprasert.

- Métodos cujos limiares de comparação se baseiam em valores máximos, mínimos e na diferença máxima entre *pixels* de imagens consecutivas [23] não consideram os valores intermediários dos *pixels* e, conseqüentemente, a susceptibilidade a pequenas variações de iluminação na cena é muito alta. Assim sendo, não serão utilizados no sistema proposto. Eliminando-se o método denominado Derivada Temporal restam apenas os seguintes métodos: Correlação Normalizada entre Blocos e o Método Estatístico de Horprasert.

- Métodos que dividem a imagem em blocos visando a exatidão na segmentação da imagem [21, 22], apresenta uma baixa resolução e, portanto, não serão utilizados no sistema proposto. Eliminado-se o método denominado Correlação Normalizada entre Blocos resta apenas o Método Estatístico de Horprasert.

- O método estatístico de Horprasert [4] utiliza um modelo computacional e dados estatísticos que levam em consideração as variações locais e globais da cena, as variações da câmera e as características de cada *pixel* na imagem em termos de valores máximos, mínimos, médios e desvio padrão. Desta forma, reúne os principais processos estatísticos utilizados separadamente nos demais métodos pesquisados. Este método ainda apresenta falhas de detecção de pontos escuros que são toleráveis no sistema proposto e, por outro lado, o processamento requerido é relativamente baixo. Sombras e pequenas variações na iluminação da cena são freqüentes na aplicação destinada ao sistema proposto e o método de Horprasert propõe-se a solucionar estes problemas. Desta forma, tendo em vista as premissas estabelecidas para o sistema proposto, o método de Horprasert [4] é o mais adequado na atualidade entre os métodos pesquisados.

Estas considerações, baseadas nas análises qualitativas dos métodos pesquisados, nortearam a escolha do método estatístico de Horprasert [4] como a técnica a ser utilizada no sistema de extração de objetos implementado neste trabalho. Trata-se de um método para detecção de objetos de uma imagem com fundo estático, que utiliza um modelo computacional cujas componentes de cor e brilho são analisadas separadamente, com robustez a pequenas variações locais e globais na imagem.

A técnica necessita de uma fase de aprendizado em que são capturados alguns quadros para determinação de parâmetros estatísticos que são utilizados na determinação automática dos limiares de comparação adequados para segmentação da imagem. Os limiares são estabelecidos pela análise de algumas características da imagem, quais sejam: imagem média, desvio padrão, distorção de brilho e distorção de cor que servem de base para a construção de histogramas com o comportamento do brilho e da cor na imagem de referência. Deste histograma, estabelecem-se os limiares para comparação e classificação da imagem.

2.4 Conclusões

Neste capítulo foram apresentados alguns métodos utilizados na segmentação de imagens. O método mais simples consiste na subtração entre quadros subseqüentes e a comparação da imagem resultante com um limiar de decisão. Métodos mais sofisticados utilizam uma imagem de referência adquirida em uma fase de aprendizado em que não há objetos na cena. Os limiares são determinados através de cálculos estatísticos, tais como: média, desvio médio, covariância, entre outros. Neste caso, a segmentação é realizada comparando-se os parâmetros da imagem do primeiro plano com os parâmetros da imagem de referência. Diferenças maiores que os limiares estabelecidos indicam que o *pixel* pertence à imagem do primeiro plano.

A determinação dos limiares de decisão utilizando parâmetros estatísticos resulta em maior robustez na segmentação da imagem. Se os parâmetros utilizados expressarem exatamente as características da cena, as variações da iluminação do ambiente e as variações da câmera, os erros de detecção serão minimizados. O método estatístico de Horprasert leva em consideração estes parâmetros.

Apresentou-se também um estudo comparativo de 10 algoritmos de segmentação de imagem e um dos resultados da simulação em Matlab do método estatístico de Horprasert [4] adaptado por Printes [7]. Mediante os resultados, observou-se que não existe algoritmo ideal capaz de sanar todos os problemas que ocorrem no ambiente e na imagem. Assim sendo, cada método é destinado a uma aplicação específica, levando-se em consideração algumas restrições.

Conforme mencionado anteriormente, a aplicação do sistema proposto é o mercado de eletrônica de consumo e as restrições de inicialização sem objetos na imagem e imagem com fundo fixo não prejudicam a aceitação no mercado consumidor. O modelo computacional e os parâmetros estatísticos utilizados no método estatístico de Horprasert [4] levam em consideração as características dos *pixels*, as variações no ambiente e na cena e as variações da câmera, o que torna o método suficientemente robusto.

No próximo capítulo é apresentada uma síntese do método estatístico de Horprasert [4], uma vez que foi detalhadamente explicado por Printes [7].

3 Solução Proposta

O algoritmo escolhido para a implementação do sistema foi apresentado por Horprasert et al [4] e adaptado para *hardware* por Printes [7] e neste capítulo são apresentados os fundamentos básicos desta técnica.

3.1 Modelo Computacional

Na Figura 3-1 mostra-se uma seqüência de imagens idênticas sendo submetidas a diferentes níveis de iluminação.

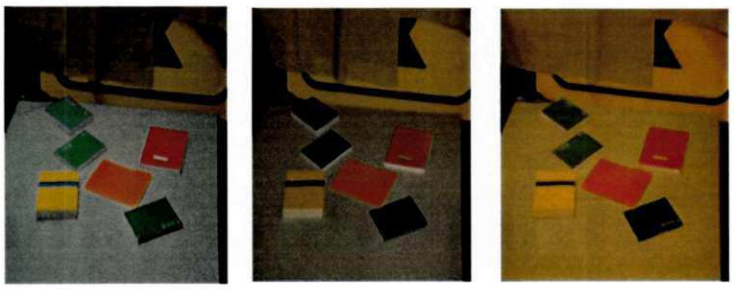


Figura 3-1: Seqüência de imagens submetidas a diferentes níveis de iluminação.

Apesar das variações de luminosidade em cada cena, a visão humana percebe as cores dos objetos sempre constantes nas três imagens. Esta característica da visão humana é denominada constância da visão. Ademais, a percepção da cor é o produto da iluminação e da reflectância espectral da superfície. Assim sendo, se a imagem de fundo for conhecida é possível detectar sombras mediante a detecção das variações de cor relacionadas com as variações de luminosidade [4], na condição de que a composição espectral da luz que ilumina a cena não mude. Estes fundamentos são a base do modelo computacional proposto [4] que consiste em analisar separadamente as componentes de cor e brilho da imagem representadas no modelo tridimensional das componentes R, G e B ilustrado na Figura 3-2.

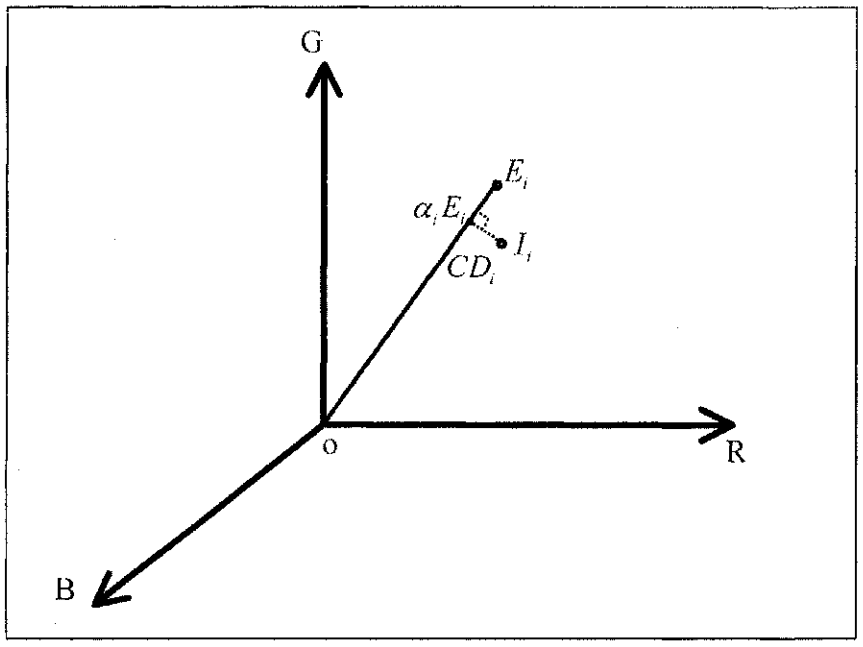


Figura 3-2: Representação do modelo computacional com as componentes RGB no modelo tridimensional.

Nesta representação, A linha $\overline{OE_i}$ que passa pela origem e pelo ponto E_i é chamada de *linha de cromaticidade esperada.* E_i é o vetor resultante entre os três eixos ortogonais e representa o valor de um *pixel i* qualquer na imagem de fundo ou referência. O módulo deste vetor representa a informação de brilho do *pixel* e o ângulo formado com os três eixos representa a cor do *pixel* da imagem de fundo. Seja I_i o mesmo *pixel* na imagem do primeiro plano. O vetor resultante deste *pixel* na imagem do primeiro plano tem a inclinação e o módulo diferente do *pixel* na imagem referência e, por conseguinte, a cor e o brilho deste *pixel* é diferente do valor esperado para imagem de referência. A extração deste *pixel*, segundo a técnica proposta, consiste em medir as "diferenças" de luminosidade e cor entre o *pixel* da imagem do primeiro plano e o *pixel* equivalente na imagem de fundo. Estas "diferenças" de luminosidade e cor são denominadas, distorção de brilho (α) e distorção de cor (CD), respectivamente [4]. A distorção de brilho é obtida da seguinte maneira [4]:

- Projeta-se o vetor I_i na linha de cromaticidade esperada. Em seguida calcula-se a menor distância entre estes vetores minimizando-se a seguinte expressão:

$$\phi(\alpha_i) = \|I_i - \alpha_i E_i\|^2 \quad \textbf{(3-1)}$$

O valor mínimo de α_i é obtido pelo cálculo da derivada da expressão para o valor igual a zero.

A distorção de cor (CD) é obtida pelo cálculo da distância ortogonal entre o vetor I_i e ponto de projeção na linha de cromaticidade segundo a seguinte expressão [4]:

$$CD_i = \|I_i - \alpha_i E_i\| \quad \textbf{(3-2)}$$

3.2 Segmentação da Imagem de Fundo

O processo de segmentação da imagem de fundo consiste nas seguintes etapas: Modelagem da Imagem de Fundo, Seleção do Nível de Limiar e Extração do Objeto [4]. A seguir é descrita cada uma destas etapas.

3.2.1 Modelagem da Imagem de Fundo

Conforme mencionado anteriormente, o algoritmo necessita de uma fase de aprendizado que consiste na aquisição de alguns quadros da imagem de fundo para obtenção de 4 parâmetros estatísticos que expressam as características da cada *pixel* na imagem de fundo e constituem o modelo individual do *pixel* nesta imagem. Os parâmetros são os seguintes:

- **Parâmetro** E_i : É a média dos valores de N quadros obtida para cada um dos *pixel*s (x,y) em cada componente de cor R,G e B, segundo a equação 3-3 [4]:

$$E_i = \left[\mu_{(i)}(C)\right] = \left[\frac{1}{N}\sum_{i=0}^{N-1} P_{i(C)}\right] \qquad \textbf{(3-3)}$$

na qual, $P_{i(C)}$ representa o valor do *pixel* i, $\mu_i(C)$ o valor médio do *pixel* i e C é cada uma das componentes de cor R, G e B.

O valor do parâmetro E_i no espaço tridimensional é representado por $E_i = \left[\mu_{(i)}(R), \mu_{(i)}(G), \mu_{(i)}(B)\right]$.

- **Parâmetro** S_i: Em função de variações na iluminação do ambiente e dos ruídos gerados pela câmera, ainda que a imagem seja estática, o mesmo *pixel* não terá valores idênticos ao longo do tempo [4]. Tendo em vista as variações de iluminação e do sensor da câmera, faz-se necessário computar as variações de cada *pixel* em N quadros em cada componente de R, G e B. Estas variações são expressas pelo desvio padrão (S_i) segundo a equação 3-4 [4]:

$$S_i = \left[\sigma_{(i)}(C)\right] = \left[\sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1}(P_{i(C)} - \mu_{(C)}(i))^2}\right] \qquad \textbf{(3-4)}$$

em que, $P_{i(C)}$ representa o valor do *pixel* i, $\sigma_{(i)}(C)$ o desvio padrão do *pixel* i e C é cada uma das componentes de cor R, G e B.

O valor do parâmetro S_i no espaço tridimensional é representado por $S_i = \left[\sigma_i(R), \sigma_i(G), \sigma_i(B)\right]$.

- **Parâmetro** α_i: Conforme mencionado, este parâmetro expressa a variação de brilho entre o *pixel* na imagem de fundo e o seu correspondente na imagem do primeiro plano. Em função das diferenças de sensibilidade dos sensores RGB da câmera, faz-se necessário um balanceamento dos *pixel*s de modo que as respostas dos sensores sejam similares para a mesma variação de intensidade de cor. Assim sendo, a distorção de brilho obtida na equação 3-5 apresenta os valores balanceados pelo desvio padrão (S_i) de cada *pixel* [4].

$$\alpha_i = \frac{\left(\frac{I_i(R)\mu_i(R)}{(\sigma_i(R))^2} + \frac{I_i(G)\mu_i(G)}{(\sigma_i(G))^2} + \frac{I_i(B)\mu_i(B)}{(\sigma_i(B))^2} \right)}{\left(\left(\frac{\mu_i(R)}{\sigma_i(R)} \right)^2 + \left(\frac{\mu_i(G)}{\sigma_i(G)} \right)^2 + \left(\frac{\mu_i(B)}{\sigma_i(B)} \right)^2 \right)} \qquad \textbf{(3-5)}$$

- **Parâmetro** CD_i: Conforme mencionado, este parâmetro expressa a variação de cor entre o *pixel* na imagem de fundo e o seu correspondente na imagem do primeiro plano. A equação 3-6 expressa o valor da distorção de cor com a equalização das respostas dos sensores da câmera efetuadas pelo desvio padrão [4].

$$CD_i = \sqrt{\left(\frac{I_i(R) - \alpha_i\mu_i(R)}{\sigma_i(R)} \right)^2 + \left(\frac{I_i(G) - \alpha_i\mu_i(G)}{\sigma_i(G)} \right)^2 + \left(\frac{I_i(B) - \alpha_i\mu_i(B)}{\sigma_i(B)} \right)^2} \qquad \textbf{(3-6)}$$

- **Parâmetro** a_i: Cada *pixel* na imagem produz diferentes valores de distorção de brilho. O parâmetro a_i é a magnitude da variação da distorção de brilho dos *pixel*s na imagem e é determinado segundo a equação 3-7 [4].

$$a_i = RMS(\alpha_i - 1) = \sqrt{\frac{\sum_{k=0}^{N-1} (\alpha_{i(k)} - 1)^2}{N}} \qquad \textbf{(3-7)}$$

- **Parâmetro** b_i : Cada *pixel* na imagem produz diferentes valores de distorção de cor. O parâmetro b_i é a magnitude da variação da distorção de cor dos *pixel*s na imagem e é determinado segundo a equação 3-8 [4].

$$b_i = RMS(CD_i) = \sqrt{\frac{\sum_{k=0}^{N-1}(CD_{i(k)})^2}{N}} \tag{3-8}$$

Uma vez determinados os parâmetros supracitados, a imagem de fundo está modelada e cada *pixel* estará representado por 4 parâmetros, quais sejam: (E_i),(S_i), (a_i) e (b_i), sendo (E_i) e (S_i) vetores de 3 componentes e (a_i) e (b_i) escalares. A próxima fase do processo de segmentação consiste na seleção dos limiares de comparação para classificação dos *pixel*s.

3.2.2 Seleção do Nível de Limiar

A fim de utilizar-se o mesmo limiar para todos os *pixel*s, necessita-se normalizar as distorções de brilho e cor. Os parâmetros utilizados para isto são as variações da distorção de brilho e cor. Seja $(\hat{\alpha}_i)$ a distorção de brilho normalizada e $(\hat{CD}_i)$ a distorção de cor normalizada, temos que:

$$\hat{\alpha}_i = \frac{\alpha_i - 1}{a_i} \tag{3-9}$$

$$\hat{CD}_i = \frac{CD_i}{b_i} \tag{3-10}$$

As distorções de brilho e cor normalizada não apresentam distribuição Gaussiana [4] e, por conseguinte, os limiares de comparação são obtidos a partir dos histogramas destas distorções. Os limiares $(\tau_{\alpha 1})$ e $(\tau_{\alpha 2})$ são utilizados para comparação

com *pixel*s com menor e maior brilho, respectivamente. O limiar (τ_{CD}) é utilizado para comparação da variação de cor entre o *pixel* da imagem de fundo e seu equivalente na imagem do primeiro plano.

Além destes limiares, adotou-se também um limite inferior mínimo para a distorção de brilho a fim de minimizar a detecção errônea de pontos escuros do objeto como sombra [4]. Ademais, pontos que possuem baixa variação de distorção de cor, geram altos valores de ($\hat{CD}_i$) e podem ocasionar erros na detecção do *pixel* na imagem do primeiro plano. Para contornar este problema foi proposta a adoção de um valor mínimo para o b_i chamado de $b_i(\min)$ [4].

3.2.3 Extração do Objeto

Na etapa de extração do objeto, cada *pixel* da imagem do primeiro plano é comparado com o seu equivalente na imagem de fundo, ou referência em termos de variação de distorção de brilho e cor. Assim sendo, o processo de extração do objeto consiste em calcular para cada *pixel* da imagem do primeiro plano as distorções de brilho e cor normalizados, utilizando o (E_i), (S_i), (a_i) e (b_i), obtidos na fase de aprendizado, e compará-los com os limiares determinados na seção 3.2.2. Se o *pixel* da imagem do primeiro plano apresentar brilho e cor similares ao seu equivalente na imagem de fundo, será classificado como fundo original (B). Caso o *pixel* na imagem do primeiro plano tenha apenas cor similar e brilho menor ou maior que seu equivalente na imagem de fundo será classificado como fundo sombreado (S) ou fundo iluminado (H), respectivamente, visto que a sombra é uma região semitransparente que mantém a cor e o padrão da superfície da imagem [4]. Se o *pixel* da imagem do primeiro plano apresenta cor diferente do limiar esperado em relação ao *pixel* na imagem de fundo será classificado como objeto (F).

Na equação 3-11 apresentam-se os limiares de comparação da imagem de fundo e classificação dos *pixel*s da imagem do primeiro plano de acordo com as distorções de brilho e cor de cada *pixel* nesta imagem [4].

$$m(i)=\begin{cases} F: C\hat{D}_i > \tau_{CD} \text{ OU } \hat{\alpha}_i < \tau_{\alpha lo}, \text{ se não} \\ B: \hat{\alpha}_i < \tau_{\alpha 1} \text{ E } \hat{\alpha}_i > \tau_{\alpha 2}, \text{ se não} \\ S: \hat{\alpha}_i < 0, \text{ se não} \\ H: \text{ outros casos} \end{cases} \quad \textbf{(3-11)}$$

3.3 Adaptações do Algoritmo Proposto

A seguir são descritas as adaptações realizadas no algoritmo proposto visando a implementação em *hardware*. Em *hardware* as operações disponíveis a um custo relativamente baixo são: soma, subtração, deslocamento para esquerda (multiplicação por dois) e deslocamento para direita (divisão por dois). As demais operações devem ser realizadas por blocos de *hardware* que executam algoritmos destas operações. As adaptações necessárias são a utilização de números em ponto fixos, mantendo a exatidão de duas casas decimais, e realizar operações de multiplicação e divisão com números de potência de dois. A exatidão de duas casas decimais é obtida pela multiplicação e divisão dos termos das equações de cálculo de parâmetros pelo fator 2^7. O fator 2^7 facilita a multiplicação e a divisão em *hardware* que pode ser implementada com registradores de deslocamento. Se fosse utilizado o fator 100 seria necessário um módulo de cálculo de divisão que é mais lento e maior em relação aos registradores de deslocamento.

3.3.1 Parâmetro E_i

O parâmetro E_i é a média dos valores de N quadros obtida para cada um dos *pixel*s (x,y) em cada componente de cor R,G e B, segundo a equação 3-3:

$$E_i = \left[\mu_{(i)}(C)\right] = \left[\frac{1}{N}\sum_{i=0}^{N-1} P_{i(C)}\right]$$

na qual, $P_{i(C)}$ representa o valor do *pixel* i, $\mu_{(i)}(C)$ o valor médio do *pixel* i e C é cada uma das componentes de cor R, G e B. Neste trabalho, utiliza-se 64 quadros para obtenção deste parâmetro. Desta forma, mantêm-se o espaço amostral necessário e facilita-se a divisão em hardware que poderá ser realizada com deslocamento à direita ou descartando-se os 8 bits menos significativos do valor acumulado.

3.3.1.1 Requisitos de Hardware

Para obtenção da média dos *pixels*, os valores temporários da soma e os valores definitivos da média são armazenados em uma memória externa. Para armazenamento temporário dos *pixels* (8 bits) nas 64 amostras teremos um número máximo de (255 x 64 = 16320) 14 bits. Cada posição de memória tem 24 bits. Portanto, para armazenar todos os valores temporários, considerando-se a resolução horizontal igual a 240 *pixels* por linha, a resolução vertical igual a 160 linhas por quadro e três posições de memória para cada pixel, são necessárias (240x160x3) 115.200 posições de memória. Para o armazenamento da média dos *pixels* utilizamos 38.400 posições, uma vez que o parâmetro E_i é um número inteiro e pode ser representado com um número binário de 8 bits, apenas uma posição é necessária para o armazenamento dos valores de R,G e B de cada *pixel.* Na Figura 3-3 mostra-se a disposição dos acumuladores e do valor definitivo da média. A razão pela qual este parâmetro está nas posições pares da memória é que nas posições ímpares são armazenados os valores do parâmetro a_i. Isto facilita durante a fase de segmentação, pois a leitura dos parâmetros é seqüencial.

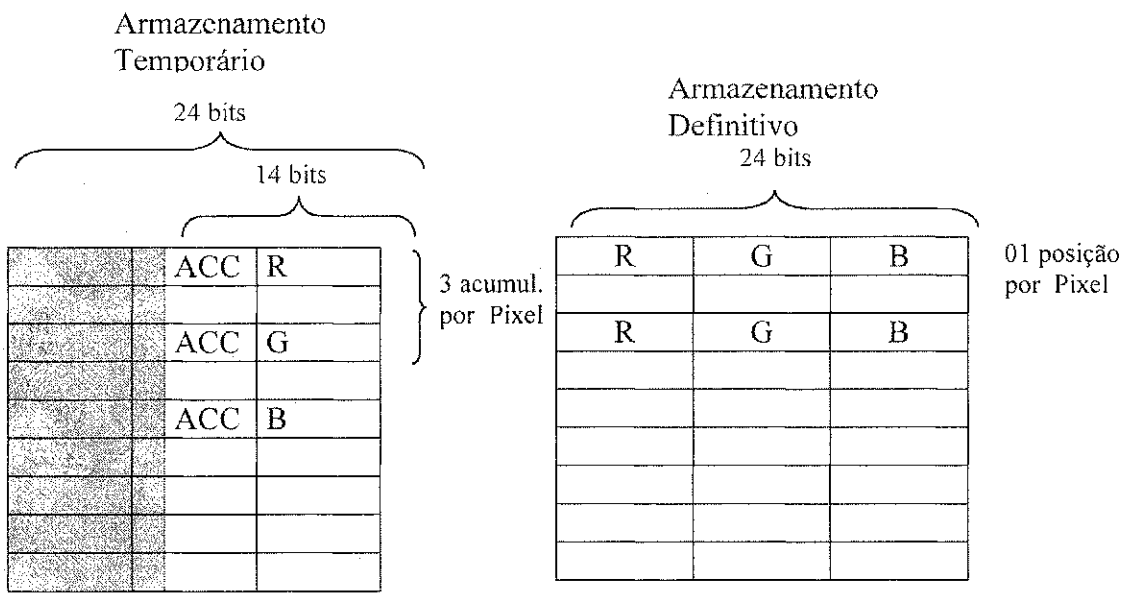


Figura 3-3: Armazenamento do parâmetro E_i.

3.3.2 Parâmetro S_i

O parâmetro S_i é formado pelas suas três componentes no espaço RGB, $S_i = [\sigma_i(R), \sigma_i(G), \sigma_i(B)]$ e representa uma estimativa da variação dos valores de 64 novas amostras medidas para cada um dos *pixels* (x,y), conforme a Equação 3-4 reescrita a seguir:

$$S_i = \left[\sigma_{(i)}(C)\right] = \left[\sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1}(P_{i(C)} - \mu_{(C)}(i))^2}\right]$$

em que, $P_{i(C)}$ representa o valor digitalizado de um determinado *pixel* i sendo C cada uma das componentes de cor R, G e B.

Originalmente [4] os desvios foram calculados para cada *pixel* da imagem. No entanto, isto demanda mais memória do sistema. Neste trabalho, calculamos o desvio de todos os *pixels* agrupados em cada componente de cor R, G e B. Uma vez que este parâmetro representa uma estimativa de variação dos sensores da câmera, esta aproximação não compromete o resultado final. Utilizou-se 27 quadros para geração do parâmetro S_i e a princípio uma resolução de 240 por 160. Com isso o número N será igual a:

N = 240x160x27 = 1.036.800 => N = 2^8 x 4050

Utilizou-se alguns artifícios matemáticos para efetuar-se as operações em ponto fixo mantendo-se a exatidão de duas casas decimais. No caso do parâmetro S_i, temos que:

$$\sigma^2_{(C)} = \frac{1}{N}\sum_{i=0}^{N-1}(P_{i(C)} - \mu_{(i)}(C))^2 \quad \text{p/ 27 quadros} \quad \Rightarrow \text{N=240x160x27 = 1.036.800}$$

$$\sigma^2_{(C)} = \frac{1}{1.036.800}\sum_{i=0}^{N-1}(P_{i(C)} - \mu_{(i)}(C))^2 \; ; \quad \text{Fatorando o valor de N, obtemos:}$$

$$\sigma^2_{(C)} = \frac{1}{4.050 \text{ x } 2^8}\sum_{i=0}^{N-1}(P_{i(C)} - \mu_{(i)}(C))^2$$

Estabelecendo-se $Dp_C = 2^8 \times \sigma^2_{(C)} = \frac{1}{4.050}\sum_{i=0}^{N-1}(P_{i(C)} - \mu_{(C)}(i))^2$, obtemos:

$$\boxed{\sigma^2_{(C)} = \frac{Dp_C}{2^8}}$$

A fim de facilitar-se as operações em hardware a seguinte aproximação pode ser realizada $N \cong 2^8 \text{ x } 4096 \Rightarrow N \cong 2^8 \text{ x } 2^{12}$, sendo facilmente implementada em hardware através de registradores de deslocamento. Esta aproximação introduz um erro de 1,1 % , para duas casas decimais, no cálculo do desvio padrão. Portanto, armazenar-se $Dp_C = 2^8 \times \sigma^2_{(C)}$, mantendo-se a exatidão numérica, e utiliza-se este valor nas operações subseqüentes. Conforme ilustrado na Figura 3-4, o erro de 1,1 % ocorre quando utiliza-se 27 quadros para o processamento do parâmetro S_i. Se o número de quadros for diferente de 27, o erro no valor do desvio padrão calculados será maior que 1,1 %.

$$Dp_C = \frac{1}{4.050}\sum_{i=0}^{N-1}(P_{i(C)} - \mu_{(C)}(i))^2 \; \therefore$$

$$\boxed{Dp_C \cong \frac{1}{2^{12}}\sum_{i=0}^{N-1}(P_{i(C)} - \mu_{(C)}(i))^2}$$

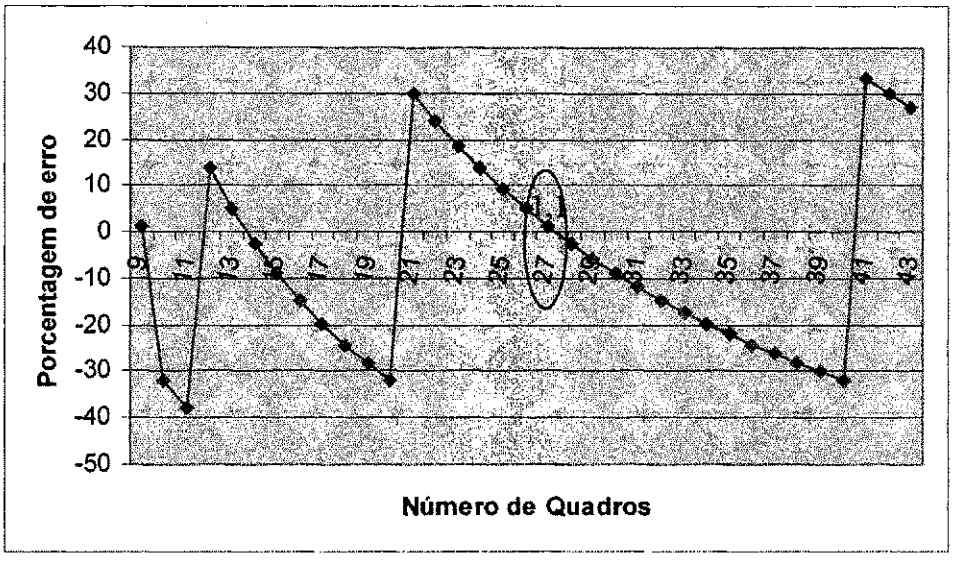


Figura 3-4: Número de quadros processados para o desvio padrão versus porcentagem de erro na operação de divisão.

3.3.2.1 Requisitos de Hardware

Conforme mencionado anteriormente o parâmetro $Dp_{(C)}$ representa o desvio padrão dos *pixels* nas componentes R, G e B. Portanto são utilizados três acumuladores para armazenar estes parâmetros. Resultados da simulação do algoritmo em MatLab mostram que os valores de S_i são inferiores a 3. Adotando-se a resolução horizontal igual a 240 elementos de imagem por linha, resolução vertical de 160 linhas e 27 quadros, obtêm-se: $(240x160x27x3^2=9.331.200)$ 24 bits para o armazenamento temporário de cada um dos 3 acumuladores para cada uma das componentes de cor. Para evitar-se divisões por zero nas operações subseqüentes estabelecemos um limite mínimo de 1,0 que corresponde a 128 com duas casas decimais. O valor máximo final para Dp_C é igual a 2.304(9331200/4050).

Portanto 3 registradores de 12 bits são necessários para armazenar os valores finais de $Dp_{(R)}$, $Dp_{(G)}$ e $Dp_{(B)}$.

3.3.3 Parâmetro αi

O parâmetro αi expressa a variação de brilho entre o *pixel* na imagem de fundo e o seu correspondente na imagem do primeiro plano. Neste caso foram analisadas duas possibilidades de rearranjo da Equação 3-5 visando a implementação em hardware. A primeira análise mantém os divisores do numerador e do denominador da Equação 3-5. A segunda elimina os divisores do numerador e do denominador que são substituídos por operações de multiplicação. A melhor opção para implementação em hardware é a segunda, pois multiplicadores utilizados neste trabalho são menores que divisores e processam os dados em 2 ciclos enquanto que divisores gastam em torno de 15 ciclos de *clock* para efetuar a operação. No capítulo 04 são mostrados os valores exatos da quantidade de células lógicas utilizadas em cada implementação.

3.3.3.1 Análise da Equação 3-5 com Divisores no Numerador e no Denominador

Considerando que o desvio padrão será obtido para os *pixels* agrupados em cada componente de cor R, G e B a equação 3-5 resulta em:

$$\alpha_i = \frac{\left(\frac{I_i(R)\mu_i(R)}{(\sigma_R)^2} + \frac{I_i(G)\mu_i(G)}{(\sigma_G)^2} + \frac{I_i(B)\mu_i(B)}{(\sigma_B)^2} \right)}{\left(\left(\frac{\mu_i(R)}{\sigma_R} \right)^2 + \left(\frac{\mu_i(G)}{\sigma_G} \right)^2 + \left(\frac{\mu_i(B)}{\sigma_B} \right)^2 \right)} \qquad \textbf{(3-12)}$$

Considerando os requisitos de duas casas decimais e visando a implementação em hardware tomemos o termo $\frac{I_i(C).\mu_i(C)}{(\sigma_C)^2}$; em que C representa cada uma das componentes R, G ou B.

Conforme obtido anteriormente, $\sigma^2_{(C)} = \frac{\mathrm{Dp_C}}{2^8}$. Substituindo-se este valor no termo $\frac{I_i(C).\mu_i(C)}{(\sigma_C)^2}$, obtemos $\frac{2^8 \times I_i(C).\mu_i(C)}{\mathrm{Dp_C}}$. Para manter-se a exatidão de duas casas decimais na operação deve-se multiplicar o termo $\frac{2^8 \times I_i(C).\mu_i(C)}{\mathrm{Dp_C}}$ por 2^7, o que resulta em:

$$Num_i(C) = \frac{2^7 \times 2^8 \times I_i(C).\mu_i(C)}{\mathrm{Dp_C}}.$$

Procedendo-se da mesma maneira com as parcelas do denominador, obtemos:

$$Den_i(C) = \frac{2^7 \times 2^8 \mu_i(C).\mu_i(C)}{\mathrm{Dp_C}}.$$

Assim sendo, a distorção de brilho será calculada por:

$$\alpha_i = \frac{\left(Num_i(R) + Num_i(G) + Num_i(B)\right) \times 2^7}{\left(Den_i(R) + Den_i(G) + Den_i(B)\right)}$$

3.3.3.2 Análise da Equação 3-5 com a Substituição dos Divisores do Numerador e do Denominador por Multiplicadores

Substituindo-se $\sigma^2_{(C)} = \frac{Dp_C}{2^8}$ na Equação 3-12, obtêm-se:

$$\alpha_i = \frac{\left(\frac{2^8 I_i(R)\mu_i(R)}{Dp_R} + \frac{2^8 I_i(G)\mu_i(G)}{Dp_G} + \frac{2^8 I_i(B)\mu_i(B)}{Dp_B} \right)}{\left(\frac{2^8 \mu_i(R)^2}{Dp_R} + \frac{2^8 \mu_i(G)^2}{Dp_G} + \frac{2^8 \mu_i(B)^2}{Dp_B} \right)}$$

Calculando o mínimo múltiplo comum no denominador e no numerador, a expressão resulta em:

$$\alpha_i = \frac{\left(\frac{2^8 I_i(R)\mu_i(R) Dp_G Dp_B}{Dp_R Dp_G Dp_B} + \frac{2^8 I_i(G)\mu_i(G) Dp_R Dp_B}{Dp_R Dp_G Dp_B} + \frac{2^8 I_i(B)\mu_i(B) Dp_R Dp_G}{Dp_R Dp_G Dp_B} \right)}{\left(\frac{2^8 \mu_i(R)^2 Dp_G Dp_B}{Dp_R Dp_G Dp_B} + \frac{2^8 \mu_i(G)^2 Dp_R Dp_B}{Dp_R Dp_G Dp_B} + \frac{2^8 \mu_i(B)^2 Dp_R Dp_G}{Dp_R Dp_G Dp_B} \right)}$$

Para manter-se a exatidão de duas casas decimais, adotada neste trabalho, dividiu-se a expressão por 2^7, o que resulta em:

$$\alpha_i = \frac{\left(\frac{2^8 I_i(R)\mu_i(R) Dp_G Dp_B}{2^7 Dp_R Dp_G Dp_B} + \frac{2^8 I_i(G)\mu_i(G) Dp_R Dp_B}{2^7 Dp_R Dp_G Dp_B} + \frac{2^8 I_i(B)\mu_i(B) Dp_R Dp_G}{2^7 Dp_R Dp_G Dp_B} \right)}{\left(\frac{2^8 \mu_i(R)^2 Dp_G Dp_B}{2^7 Dp_R Dp_G Dp_B} + \frac{2^8 \mu_i(G)^2 Dp_R Dp_B}{2^7 Dp_R Dp_G Dp_B} + \frac{2^8 \mu_i(B)^2 Dp_R Dp_G}{2^7 Dp_R Dp_G Dp_B} \right)}$$

Efetuando-se a simplificação do numerador e do denominador, obtêm-se:

$$\alpha_i = \frac{\left(\frac{2I_i(R)\mu_i(R)Dp_G Dp_B}{Dp_R Dp_G Dp_B} + \frac{2I_i(G)\mu_i(G)Dp_R Dp_B}{Dp_R Dp_G Dp_B} + \frac{2I_i(B)\mu_i(B)Dp_R Dp_G}{Dp_R Dp_G Dp_B} \right)}{\left(\frac{2\mu_i(R)^2 Dp_G Dp_B}{Dp_R Dp_G Dp_B} + \frac{2\mu_i(G)^2 Dp_R Dp_B}{Dp_R Dp_G Dp_B} + \frac{2\mu_i(B)^2 Dp_R Dp_G}{Dp_R Dp_G Dp_B} \right)}$$

Dividindo-se toda a expressão por 2, obtêm-se:

$$\alpha_i = \frac{\left(\frac{I_i(R)\mu_i(R)Dp_G Dp_B}{Dp_R Dp_G Dp_B} + \frac{I_i(G)\mu_i(G)Dp_R Dp_B}{Dp_R Dp_G Dp_B} + \frac{I_i(B)\mu_i(B)Dp_R Dp_G}{Dp_R Dp_G Dp_B} \right)}{\left(\frac{\mu_i(R)^2 Dp_G Dp_B}{Dp_R Dp_G Dp_B} + \frac{\mu_i(G)^2 Dp_R Dp_B}{Dp_R Dp_G Dp_B} + \frac{\mu_i(B)^2 Dp_R Dp_G}{Dp_R Dp_G Dp_B} \right)}$$

Para manter-se a exatidão de duas casas decimais na operação de divisão, multiplica-se o numerador por 2^7 e a expressão final é:

$$\alpha_i = \frac{2^7 \left(I_i(R)\mu_i(R)Dp_G Dp_B + I_i(G)\mu_i(G)Dp_R Dp_B + I_i(B)\mu_i(B)Dp_R Dp_G \right)}{\left(\mu_i(R)^2 Dp_G Dp_B + \mu_i(G)^2 Dp_R Dp_B + \mu_i(B)^2 Dp_R Dp_G \right)}$$

Esta expressão é implementada em hardware por razões de otimização de células e velocidade de processamento. A seguir, analisam-se os requisitos de hardware para implementação desta expressão.

3.3.3.3 Requisitos de Hardware

Para efetuar-se a multiplicação de $I \times \mu$ é necessário um multiplicador 8x8 bits. Para multiplicar-se $Dp_{(C)}$ x $Dp_{(C)}$ é necessário um multiplicador de 12x12bits. O resultado de $Dp_{(C)}$ x $Dp_{(C)}$ será multiplicado pelo resultado de $I \times \mu$ por um multiplicador de 24x16bits. Substituindo-se os valores de I_i=255, $\mu = 255$ e $Dp_{(C)} = 2304$ na expressão de α_i, obtemos um divisor máximo de:

$$\alpha_i = \frac{418,7158466x10^{12}}{3,271217552x10^{12}} = \frac{49bits}{42bits}$$

No capítulo 4 é apresentado o diagrama em blocos para implementação desta expressão.

3.3.4 Parâmetro *CDi*

Conforme mencionado, este parâmetro expressa a variação de cor entre o *pixel* na imagem de fundo e o seu correspondente na imagem do primeiro plano. Considerando que o desvio padrão será obtido para os *pixels* agrupados em cada componente de cor R, G e B a equação 3-6 resulta em:

$$CD_i = \sqrt{\left(\frac{I_i(R)-\alpha_i\mu_i(R)}{\sigma_R}\right)^2+\left(\frac{I_i(G)-\alpha_i\mu_i(G)}{\sigma_G}\right)^2+\left(\frac{I_i(B)-\alpha_i\mu_i(B)}{\sigma_B}\right)^2} \qquad (3\text{-}13)$$

A expressão anterior consiste na extração da raiz quadrada da soma de 3 parcelas. Analisando uma das parcelas, considerando-se o seguinte:

$$CD_i(C) = \frac{\left(I_i(C)-\alpha_i\mu_i(C)\right)^2}{\left(\sigma_{(C)}\right)^2}$$

Substituindo-se na expressão da distorção de cromaticidade, obtêm-se:

$$CD_i = \sqrt{CD_i(R)+CD_i(G)+CD_i(B)}$$

Conforme procedemos com a expressão da distorção de brilho, com a expressão da distorção de cor também foram efetuados alguns arranjos matemáticos a fim de manter-se a exatidão de duas casas decimais nos cálculos.

Para a operação de $I_i(C)-\alpha_i\mu_i(C)$, o número $I_i(C)$ deve entrar nos cálculos multiplicado por 2^7. Limitou-se o resultado desta operação em 32, uma vez que não é

necessário avaliar-se *pixels* com distâncias superiores a este valor, eliminando-se a necessidade de *hardware* adicional. Substituindo-se $\sigma^2_{(C)} = \frac{\mathrm{Dp_C}}{2^8}$ em $\frac{\left(I_i(C) - \alpha_i \mu_i(C)\right)^2}{\left(\sigma_C\right)^2}$, obtemos: $\frac{2^8 \times \left(I_i(C) - \alpha_i \mu_i(C)\right)^2}{Dp_C}$. No numerador têm-se um número com 4 casas decimais, e para manter-se o formato adotado neste trabalho dividiu-se o resultado por 2^7, o que resulta em $\frac{2 \times \left(I_i(C) - \alpha_i \mu_i(C)\right)^2}{Dp_C}$. A operação final consiste na extração da raiz quadrada de acordo com $\sqrt{CD_i(R) + CD_i(G) + CD_i(B)}$. Para manutenção de 2 casas decimais multiplicamos a expressão por 2^7 e a operação final realizada será $\sqrt{\left(CD_i(R) + CD_i(G) + CD_i(B)\right) \times 2^7}$.

3.3.4.1 Requisitos de Hardware

Para calcular-se o termo $\alpha_i . \mu_i(C)$ é necessário um multiplicador de 10x8 bits, visto que o α_i é um número de 10 bits com exatidão de 2 casas decimais e $\mu_i(C)$ é um número de 8 bits sem casas decimais.

Substituindo-se os valores de $Dp_{(C)} = 2304$, $(I_i(C) - \alpha_i \mu_i(C)) = 4096$ e somando-se as 3 parcelas da expressão da distorção de cor, obtemos o divisor necessário para a operação que tem:

$$CDi_{(c)} = \frac{6(4096)^2}{2304} = \frac{27bits}{12bits}$$

O bloco de cálculo da raiz quadrada deverá extrair a raiz quadrada de um número com no máximo 27 bits, conforme o cálculo a seguir.

$$CDi_{(c)} = \frac{2^7 6(4096)^2}{128} = 100,66x10^6 = 27bits$$

Portanto, o parâmetro CD_i será um número, no máximo, entre 13 e 14 bits. Nas análises a seguir considera-se o parâmetro CD_i com no máximo 14 bits para facilitar-se a obtenção dos requisitos de *hardware* dos demais parâmetros.

3.3.5 Parâmetros b_i e $C\hat{D}_i$

O parâmetro b_i é a magnitude da variação da distorção de cor dos *pixels* em 64 quadros e é determinado segundo a equação 3-8 reescrita a seguir para melhor entendimento do texto.

$$b_i = RMS(CD_i) = \sqrt{\frac{\sum_{k=0}^{N-1}(CD_{i(k)})^2}{N}}$$

Para acumular-se o valor de $(CD_{i(k)})^2$ são necessários acumuladores de $\left(2^{14}\right)^2 x64 = 17,1798x10^9 = 34bits$. A memória externa disponível tem apenas 24 bits e, portanto não se pode acumular este valor. Desta maneira, acumula-se apenas o valor de CD_i. Esta simplificação não compromete o resultado final visto que o parâmetro b_i é apenas um parâmetro de normalização. Assim sendo, para acumular o valor do CD_i em 64 quadros utilizaremos $2^{14}x64 = 1,04857x10^6 = 20bits$.

O parâmetro $C\hat{D}_i$ é a distorção de cor normalizada obtida pela razão entre a distorção de cor e a magnitude da variação da distorção de cor. O divisor necessário para processamento deste parâmetro será de:

$$\frac{2^{14}x128}{2^{14}} = \frac{21bits}{14bits}$$

3.3.6 Parâmetro a_i e $\hat{\alpha}_i$

O parâmetro a_i é a magnitude da variação da distorção de brilho dos *pixels* em 64 quadros e é determinado segundo a equação 3-7 reescrita a seguir para melhor entendimento do texto.

$$a_i = RMS(\alpha_i - 1) = \sqrt{\frac{\sum_{k=0}^{N-1}(\alpha_{i(k)} - 1)^2}{N}}$$

A operação $(\alpha_i - 1)^2$ resultará em número com exatidão de 4 casas decimais. Dividi-se o resultado de $(\alpha_i - 1)^2$ por 2^7 antes de registrar no acumulador (memória externa). No cálculo final é necessário efetuar a extração da raiz quadrada. Para manter-se a exatidão de 2 casas decimais multiplicamos por 2^7. O valor da extração da raiz quadrada será no máximo igual a:

$$a_{i(\max)} = \sqrt{\frac{N \times \left[\frac{\left(\alpha_{i(\max)} - 1 * 2^7\right)^2}{2^7}\right]}{N}} \times 2^7$$

Substituindo-se os valores, obtemos:

$$\sqrt{\frac{64 \times \left[\frac{(512 - 128)^2}{128}\right]}{64}} \times 128 = \sqrt{147456} = 10 bits$$

O parâmetro $\hat{\alpha}_i$ é a distorção de brilho normalizada obtida pela razão entre a distorção de brilho e a magnitude da variação da distorção de brilho. O divisor necessário para processamento deste parâmetro será de:

$$\widehat{\alpha_i} = \frac{\alpha_i \times 128}{a_{i(\max)}} = \frac{512 \times 128}{384} = \frac{16}{9} bits$$

3.4 Conclusões

Neste capítulo, apresentou-se sucintamente o método estatístico de Horprasert. Trata-se de um método que utiliza um modelo computacional em que as informações de brilho e cor são analisadas separadamente e computadas estatisticamente.

O método se propõe a resolver as variações locais e globais na imagem. Descreveu-se também algumas adaptações nas equações visando a implementação em hardware. No próximo capítulo será apresentado o sistema e as etapas de implementação do método proposto utilizando a plataforma para sistemas de imagens [7].

4 Sistema para Extração de Objetos

Neste capítulo é apresentada a plataforma de desenvolvimento [7] e os blocos de *hardware* utilizados para implementação da solução proposta. O fluxo de projeto utilizado na implementação do sistema consiste de 06 etapas, quais sejam: Definição do Sistema, Modelagem em HDL (*hardware description language*) , Verificação Comportamental e Estrutural , Síntese e Leiaute. Cada uma dessas etapas também é descrita neste capítulo.

4.1 Plataforma de Desenvolvimento

A plataforma de desenvolvimento utilizada no sistema para extração de objetos é a GPIP-01 (*General Purpose Image Processor*) implementada e apresentada por Printes [7]. Esta plataforma contém os blocos mínimos para elaboração e validação de sistemas para processamento de imagens, quais sejam: Estágio de Aquisição de Imagens, Estágio de Armazenamento, Estágio de Processamento, Estágio de Apresentação, todos elaborados com circuitos integrados dedicados para a respectiva função. O diagrama em blocos da plataforma é mostrado na Figura 4-1 e descrito a seguir.

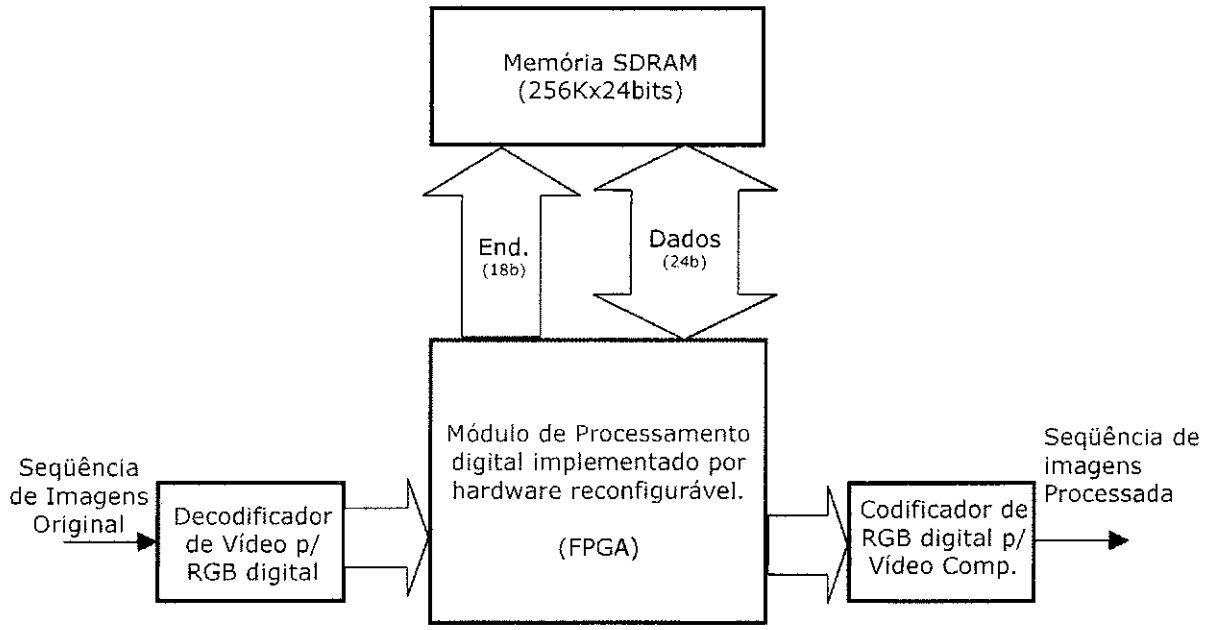


Figura 4-1: Diagrama em blocos da Plataforma GPIP01.

- Estágio de aquisição de imagens: É composto por um decodificador de vídeo que converte o sinal de vídeo composto para o formato RGB digital 24 bits com um *clock* de *pixel* de 13.5 MHz. O circuito integrado utilizado foi o SAA7111 fabricado pela Philips Semiconductors.

- Estágio de Processamento: Este estágio contém um circuito integrado reconfigurável, FPGA da família APEX -EP20K200EQC240-2X, da Altera Corporation. Neste estágio estão as interfaces de controle dos estágios de aquisição de imagens, memória e apresentação. Tais interfaces ocupam cerca de 13% dos elementos lógicos disponíveis no dispositivo [7]. Portanto, restam 87% de elementos lógicos para desenvolvimento do sistema destinado à extração de objetos.

- Estágio de Armazenamento: É composto por um banco de memória de 264Kx24 bits implementado com 04 circuitos integrados de memória UT6164C32 fabricada pela UTRON. Este banco de memórias será utilizado para armazenamento temporário e definitivo dos parâmetros calculados no algoritmo proposto para o sistema de extração de objetos.

- Estágio de Apresentação: Este estágio consiste de um codificador vídeo que converte o sinal em formato RGB digital 24 bits para vídeo analógico, possibilitando a exibição dos resultados do processamento. O circuito integrado utilizado foi o THS8134B fabricado pela Texas Instruments composto por três conversores D/A de 8 bits para aplicações de vídeo.

4.2 Definição do Sistema

A definição do sistema consiste em "traduzir" as especificações do sistema em diagrama em blocos. Neste trabalho as especificações são as equações do algoritmo de Horprasert [4] que foram adaptadas por Printes [7] para implementação em *hardware*. Nas próximas seções, apresenta-se uma análise dos requisitos de operação e descrevem-se os blocos necessários para implementação do sistema.

4.2.1 Análise Preliminar dos Requisitos de Hardware

A fase inicial do projeto consistiu em validar as adaptações para implementação em *hardware* sugeridas por Printes [7] no Matlab para validação, uma vez que o sistema executa cabalmente cada equação do algoritmo. Uma vez validado o algoritmo com as adaptações necessárias, fez-se a definição dos blocos de *hardware* necessários para cada parâmetro do algoritmo. Printes [7] propôs teoricamente os blocos e as máquinas de estados para implementação do algoritmo adaptado. Inicialmente, estimou-se a ocupação de

elementos lógicos, sobretudo, dos blocos de divisão, multiplicação e raiz quadrada em cada bloco proposto por Printes [7]. Os multiplicadores foram criados no Quartus [27] a partir de funções disponíveis. Os divisores implementados foram baseados no método de divisão de números binários que pode ser encontrado em [28]. O bloco para extração de raiz quadrada foi implementado a partir de um algoritmo em C apresentado por Tammiska [29]. Para obtenção desta estimativa, os blocos foram implementados e sintetizados fisicamente para a FPGA[3] família APEX -EP20K200EQC240-2X. Os resultados estão ilustrados na Tabela 4-1.

[3] FPGA: *Field Programmable Gate Array* – Dispositivo com um conjunto de portas programáveis.

Tabela 4-1: Estimativa de elementos lógicos necessários para implementação dos blocos de cálculo do sistema de extração de objetos.

Blocos de Cálculo (% Elementos Lógicos)	**Quantidade de Blocos de Cálculo para cada Parâmetro**					
	Gerador S_i	**Gerador** α_i	**Gerador** CD_i	**Gerador** a_i **e** b_i	**Gerador** $\hat{CD}_i$	**Gerador** $\hat{\alpha}_i$
Multiplicador 8x8 (1%)	1	3	-	-	-	-
Multiplicador 10x8 (1%)	-	-	3	-	-	-
Multiplicador 13x13 (3%)	-	-	3	-	-	-
Multiplicador 10x10 (3%)	-	-	-	1	-	-
Divisor 31/12 (9%)	-	3	-	-	-	-
Divisor 33/26 (14%)	-	1	-	-	-	-
Divisor 27/12 (8%)	-	-	3	-	-	-
Raiz Quadrada (4%)	-	-	1	1	-	-
Divisor 20/10 (7%)	-	-	-	-	-	1
Divisor 28/14 (8%)	-	-	-	-	1	-
Soma Parcial de Elementos Lógicos (%)	1	44	40	5	8	7
SOMA TOTAL DE ELEMENTOS LÓGICOS: 105%						

Conforme os resultados, a quantidade de elementos lógicos necessários para implementar os blocos de cálculos é superior ao disponível no dispositivo da plataforma GPIP01. Considerando-se ainda as máquinas de estados para controle de cada bloco do sistema, certamente este número crescerá ainda mais. Na seção 4.2.2 é apresentada a nova arquitetura do sistema implementado neste trabalho.

As implementações destes blocos iniciais também mostraram o número de ciclos de *clock* necessários para execução de cada operação. Os multiplicadores processam os dados em 2 ciclos, um divisor (27/12) otimizado em velocidade processa uma operação em 14 ciclos de *clock* e o bloco de extração de raiz quadrada processa a operação em 19 ciclos de *clock*. Estes dados foram utilizados para determinação do número de ciclos necessários para processamento de cada *pixel*. Considerando-se, por exemplo, o cálculo do parâmetro CD_i em que há uma divisão e uma raiz quadrada, o número de ciclos somente para estas duas operações seria igual a 33. Desta maneira, foi estabelecido 54 ciclos de *clock* para o processamento de cada *pixel*. Na Tabela 4-2 mostram-se alguns valores de freqüência de *clock* do sistema em função das resoluções horizontal, vertical e do tempo de processamento considerando-se 54 ciclos de operação para cada pixel.

Tabela 4-2: Freqüência de operação do sistema em função das resoluções horizontal e vertical da imagem e do tempo de processamento de cada *pixel* considerando 54 ciclos de operação.

Resolução Vertical (Linhas/Quadro)	Resolução Horizontal (Pixel/Linha)	Tempo de Processamento (μs)	Freqüência *Clock* Sistema (MHz)
480	360	63,4	306
240	360	126,8	153
160	360	190,2	102
160	240	190,2	68
160	180	190,2	51
160	144	190,2	40
160	120	190,2	34

O tempo de processamento é o tempo disponível para o processamento de todos os *pixels* de uma linha que deve ser concluído antes da aquisição da próxima linha. O tempo de processamento e a freqüência de *clock* são obtidos pelas equações 4-2 e 4-1, respectivamente.

As resoluções horizontal e vertical na Tabela 4-2 foram obtidas a partir das especificações do decodificador de vídeo que são 720 pixels/linha e 480 linhas/quadro. Os valores 240 linhas/quadro e 360 pixels/linha correspondem a 480/2 e 720/2, respectivamente. Mediante os resultados da Tabela 4-2 adotamos a freqüência inicial do sistema igual a 68MHz, visto que a freqüência operação máxima do dispositivo da plataforma é 150MHz e a memória da plataforma GPIP-01 tem uma especificação de freqüência máxima igual a 70MHz. Outro aspecto importante para determinação da freqüência do sistema é o número de elementos lógicos utilizados. Quanto maior for a quantidade de elementos lógicos, maiores são as dificuldades para elaboração do leiaute e com isso as ligações entre as estruturas tornam mais longas e, por conseguinte, os atrasos

dos sinais são maiores. Assim sendo, em dispositivos de lógica programável, quanto maior a porcentagem de utilização dos elementos lógicos, menor será a freqüência de operação em função dos atrasos internos. Estabeleceu-se a freqüência de 68MHz como ponto de partida para o desenvolvimento do sistema, no entanto, na etapa de simulação com atrasos, constatou-se que com esta freqüência ocorrem atrasos maiores que ½ ciclo de clock e com isso a freqüência do sistema foi reduzida. Isto será mostrado na seção 4.6.11.

$$Fclock = \frac{1}{\frac{Tprocessamento}{N.^{\circ}ciclos \times \mathrm{Re}sH}} \tag{4-1}$$

$$Tprocessamento = \frac{480 \times 63,4\mu}{\mathrm{Re}sV} \tag{4-2}$$

4.2.2 Definição dos Blocos do Sistema

A solução implementada consiste na distribuição do sistema em duas FPGA conforme mostrado na Figura 4-2. Ainda assim foram feitas algumas mudanças no sistema inicialmente proposto a fim de atendermos aos critérios de processamento e freqüência de operação conforme descrito nas seções 4.2.2.1 a 4.2.2.16 e 4.6.11.

O sistema implementado na FPGA 01 é composto pelos seguintes blocos: Interface I2C, Interface Decodificador de Vídeo, Multiplexador de Sinais de Controle, Interface de Acesso à Memória, Gerador de Parâmetros $E_i\_S_i$, Gerador de Parâmetros $\alpha_i\_a_i$, Inicialização da Memória e *Buffer* de Parâmetros. A FPGA 01 é responsável pela captura dos *pixels* oriundos do decodificador de vídeo e pelo processamento dos parâmetros E_i, S_i,

a_i, α_i e $\hat{\alpha}_i$. Há também um *buffer* interno que é utilizado para armazenar os parâmetros que são transferidos para a FPGA 02.

O sistema implementado na FPGA 02 é composto pelos seguintes módulos: Controlador da Interface de Acesso a Memória, Registradores de Parâmetros, *Buffer* de Parâmetros, Gerador de Parâmetros $C\hat{D}_i\_b_i$, Gerador de Histograma e Limiar, Segmentação, Apresentação do Objeto, Interface Codificador Vídeo, Transferência de Parâmetros e Interface de Acesso à Memória. Na FPGA 02 são processados os parâmetros CD_i, $C\hat{D}_i$ e b_i. Além disso, são obtidos o histograma do parâmetro $C\hat{D}_i$, é determinado o limiar de comparação utilizado para extração de objetos, é realizada a segmentação da imagem e também a apresentação do objeto extraído no codificador de vídeo. A plataforma de desenvolvimento utilizada no sistema de extração de objetos é mostrada na Figura 4-3.

Nas seções 4.2.2.1 a 4.2.2.16 é descrito o funcionamento básico de cada bloco implementado no sistema de extração de objetos.

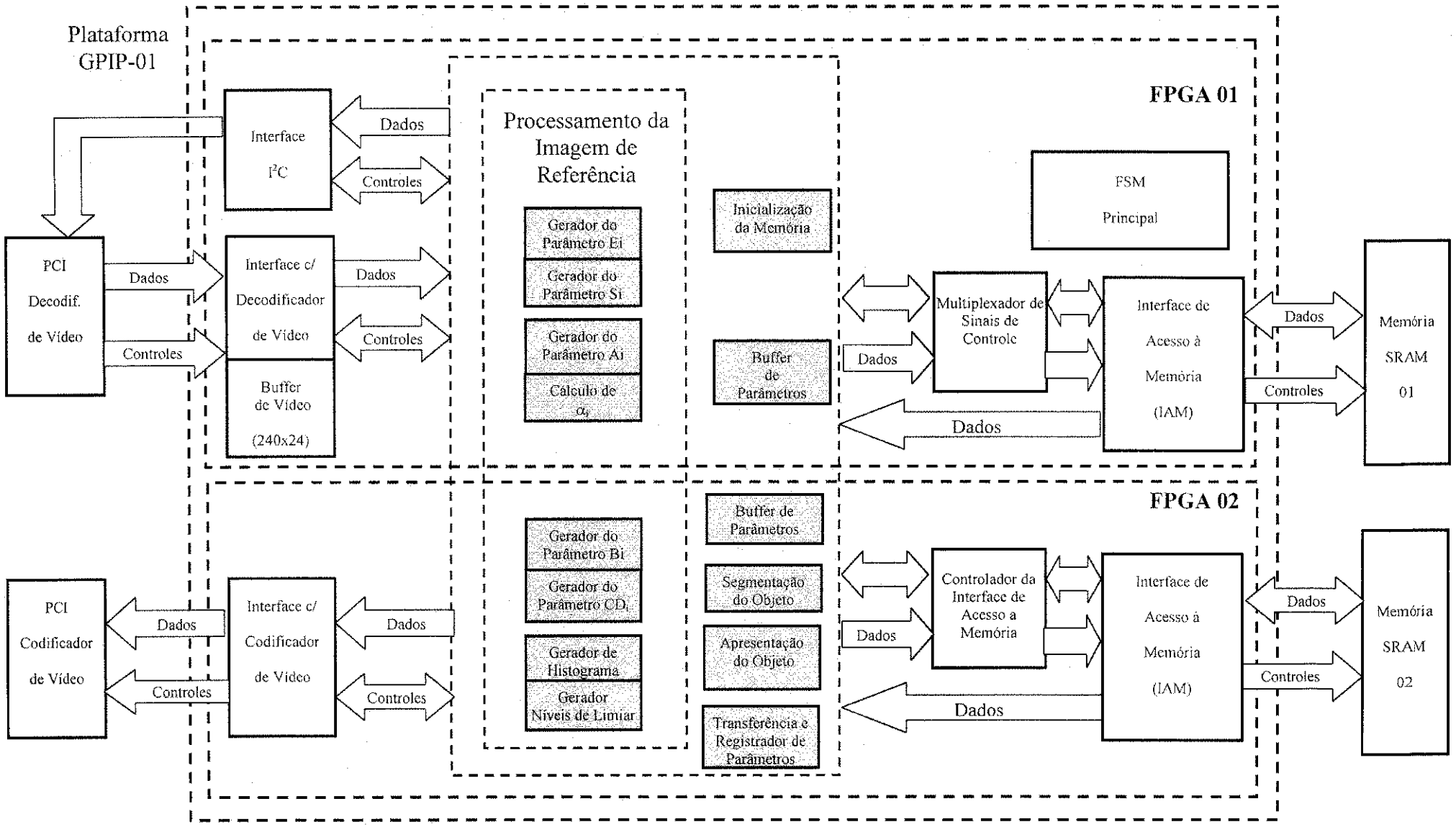


Figura 4-2: Diagrama em blocos geral do sistema de extração de objetos implementado em *hardware*

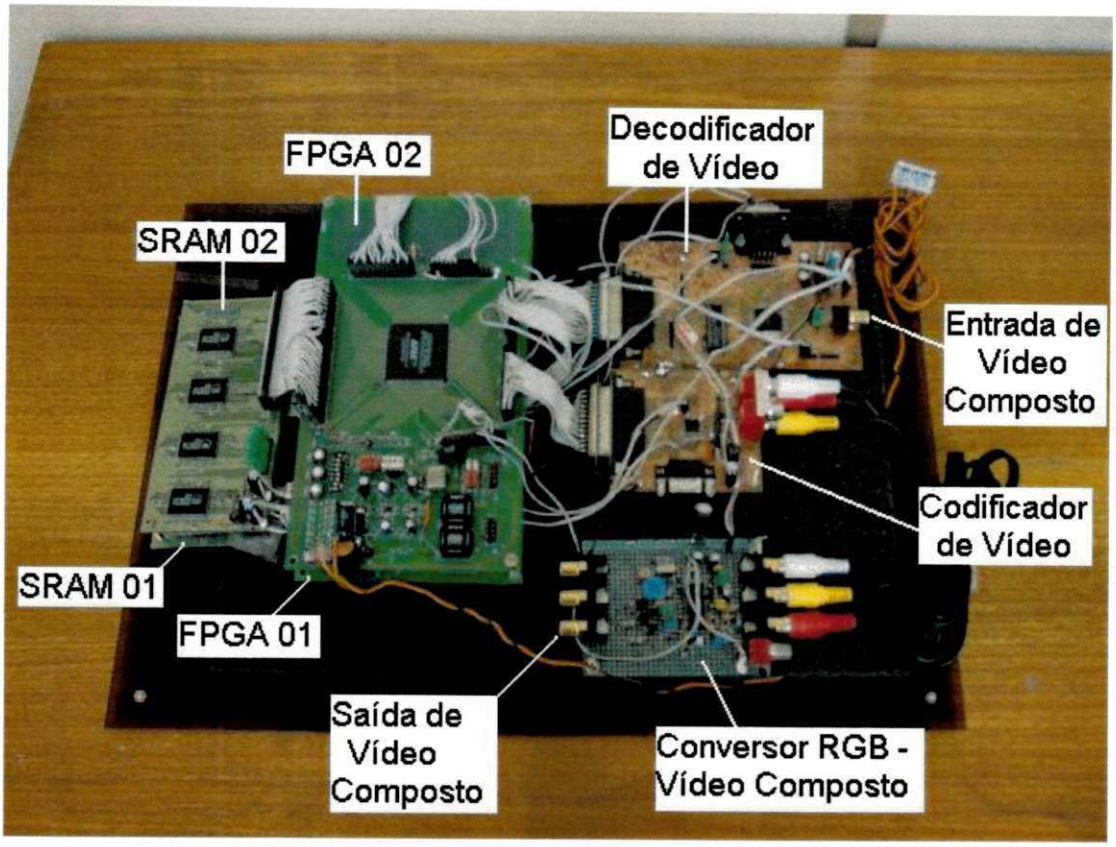


Figura 4-3: Plataforma utilizada no sistema de extração de objetos.

4.2.2.1 Fsm_Principal

A Fsm_Principal é composta basicamente por uma máquina de estados e um contador de quadros. A Fsm_Principal é o bloco responsável por ordenar as etapas de aprendizado e segmentação do sistema, habilitando cada bloco do sistema e verificando o número de quadros processados. As etapas de processamento realizadas pela Fsm_Principal estão enumeradas a seguir.

- **Fase de Aprendizado**: Na fase de aprendizado, uma seqüência de quadros da imagem de fundo, sem objetos do primeiro plano, é adquirida para obtenção dos parâmetros estatísticos que são utilizados durante a fase de segmentação. Conforme mencionado anteriormente, os parâmetros são os seguintes: imagem média, desvio padrão, variação média da distorção de brilho, variação média da distorção de cor, histograma da distorção de cor e determinação do limiar de comparação.

- **Inicialização da Memória**: A primeira operação comandada pela Fsm_Principal é a inicialização da memória externa utilizada para o armazenamento temporário e definitivo da média, variação média da distorção de brilho e variação média da distorção de cor. Quando todas as posições da memória estiverem com zeros, o bloco de inicialização sinaliza para a Fsm_Principal que inicia a próxima etapa.

- **Geração do Parâmetro** E_i : A partir deste ponto, a Fsm_Principal sempre inicializa a interface com o decodificador de vídeo no início e ao término do processamento de um quadro. Nesta etapa são calculados os valores médios dos *pixels* em 64 quadros. Primeiro, a Fsm_Principal habilita o Gerador de Parâmetros $E_i\_S_i$ para acumular a soma dos valores dos *pixels* na memória. O Gerador de Parâmetros $E_i\_S_i$ a cada quadro processado sinaliza para a Fsm_Principal. A Fsm_Principal reinicia o ciclo até que 64 quadros tenham sido processados. Em seguida, a Fsm_Principal habilita o Gerador de Parâmetros $E_i\_S_i$ para calcular a média dos valores armazenados na

memória e armazená-los definitivamente na memória. Ao término do cálculo da média o Gerador de Parâmetros $E_i\_S_i$ sinaliza para Fsm_Principal que inicia a etapa de aquisição do desvio padrão.

- **Geração do Parâmetro** S_i: Nesta etapa são calculados os desvios padrões dos *pixels* para cada componente de cor R, G e B em 27 quadros. Para o cálculo do desvio padrão são utilizados os valores médios dos *pixels* previamente calculados e disponíveis na memória externa. Primeiro, a Fsm_Principal habilita o Gerador de Parâmetros $E_i\_S_i$ para acumular, em registradores internos, os valores parciais da operação (desvio padrão). O Gerador de Parâmetros $E_i\_S_i$ ao término do acúmulo de um quadro sinaliza para Fsm_Principal que reinicia o ciclo até que 27 quadros tenham sido processados. Em seguida, a Fsm_Principal habilita o Gerador de Parâmetros $E_i\_S_i$ para calcular o $Dp_{(R)}$, $Dp_{(G)}$ e $Dp_{(B)}$ finais. Ao término deste cálculo o Gerador de Parâmetros $E_i\_S_i$ sinaliza para a Fsm_Principal que inicia as etapas de aquisição dos valores da variação média das distorções de brilho e de cor.

- **Geração dos Parâmetros** a_i e b_i: Nesta etapa são calculados os valores da variação média das distorções de brilho e de cor. Os valores de a_i são armazenados na memória externa conectada à FPGA 01 e os valores de b_i na memória externa conectada à FPGA 02. Para o cálculo do a_i utilizamos os valores de $Dp_{(R)}$, $Dp_{(G)}$, $Dp_{(B)}$, os valores da média dos *pixels* previamente calculados e os valores dos *pixels* correntes. Estes parâmetros também são utilizados para o cálculo do b_i juntamente com o valor de α_i que é calculado nesta etapa. O Gerador de Parâmetros $C\hat{D}_i\_b_i$ está na FPGA 02, portanto os parâmetros $Dp_{(R)}$, $Dp_{(G)}$, $Dp_{(B)}$, E_i, $\hat{\alpha}_i$ e α_i são transferidos para a FPGA 02 para processamento do b_i. Inicialmente, a Fsm_Principal habilita os blocos Gerador de Parâmetros $\hat{\alpha}_i\_a_i$ e Gerador de Parâmetros $C\hat{D}_i\_b_i$ para calcularem e armazenarem os resultados parciais nas memórias externas. Os

blocos Gerador de Parâmetros $\hat{\alpha}_i\_a_i$ e Gerador de Parâmetros $\hat{CD}_i\_b_i$ sinalizam para a Fsm_Principal o término do processamento de um quadro completo. Em seguida, a Fsm_Principal habilita-os novamente e reinicia o ciclo até que 64 quadros tenham sido processados. Ao término dos 64 quadros, a Fsm_Principal habilita os blocos Gerador de Parâmetros $\hat{\alpha}_i\_a_i$ e Gerador de Parâmetros $\hat{CD}_i\_b_i$ para calcularem o resultado final da operação e armazenarem os definitivos nas memórias externas. Ao término desta operação, os blocos Gerador de Parâmetros $\hat{\alpha}_i\_a_i$ e Gerador de Parâmetros $\hat{CD}_i\_b_i$ sinalizam para a Fsm_Principal que inicia as etapas de geração do histograma e determinação do limiar de comparação.

- **Geração do Histograma e Determinação do Limiar de Comparação**: Nesta etapa é construído um histograma da distorção de cor normalizada ($\hat{CD}_i$) e a partir deste histograma determina-se o limiar de comparação utilizado para fase de segmentação do objeto. Para calcular-se a distorção de cor normalizada são utilizados os valores da distorção de cor dos *pixels* correntes e os valores de b_i calculados previamente e armazenados na memória externa. A Fsm_Principal habilita os blocos Gerador de Parâmetros $\hat{\alpha}_i\_a_i$, Gerador de Parâmetros $\hat{CD}_i\_b_i$ e o Gerador de Histograma e Limiar para iniciarem o processamento de 64 quadros. Ao término do processamento de 64 quadros, a Fsm_Prinicipal habilita o Gerador de Histograma e Limiar para calcular o limiar de comparação baseado na distribuição dos valores de $\hat{CD}_i$ previamente calculados e armazenados em um *buffer* interno. Quando o limiar é determinado, a Fsm_Prinicipal iniciará a segunda fase de processamento do sistema, isto é, a fase de segmentação.

- **Fase de Segmentação**: Na fase de segmentação, os blocos Gerador de Parâmetros $\hat{\alpha}_i\_a_i$, Gerador de Parâmetros $\hat{CD}_i\_b_i$, Segmentação, Apresentação do Objeto e Interface com Codificador de Vídeo são constantemente habilitados para efetuarem a extração e apresentação do objeto da imagem do primeiro plano.

4.2.2.2 Interface I2C

A função desta interface é configurar o decodificador de vídeo para que ele disponibilize um sinal de vídeo digitalizado para o formato RGB. O diagrama em blocos desta interface é mostrado na Figura 4-4. A FSM_Master contém as palavras de controle para programação do decodificador de vídeo. Durante a inicialização, ela começa a transferir as palavras de configuração para o registrador de deslocamento e habilita a Fsm_i2c que gera o protocolo I2C (*start bit*, *clock*, dados e *stop bit*) de acordo com os sinais de controle da entrada. No início, a Fsm_i2c gera um *start bit* e começa a transferir os dados do registrador de deslocamento e ao final sinaliza para a FSM_Master. Quando todas as palavras tiverem sido transferidas a FSM_Master sinaliza fim de dados para Fsm_i2c que gera o *stop bit*, finalizando a programação da interface.

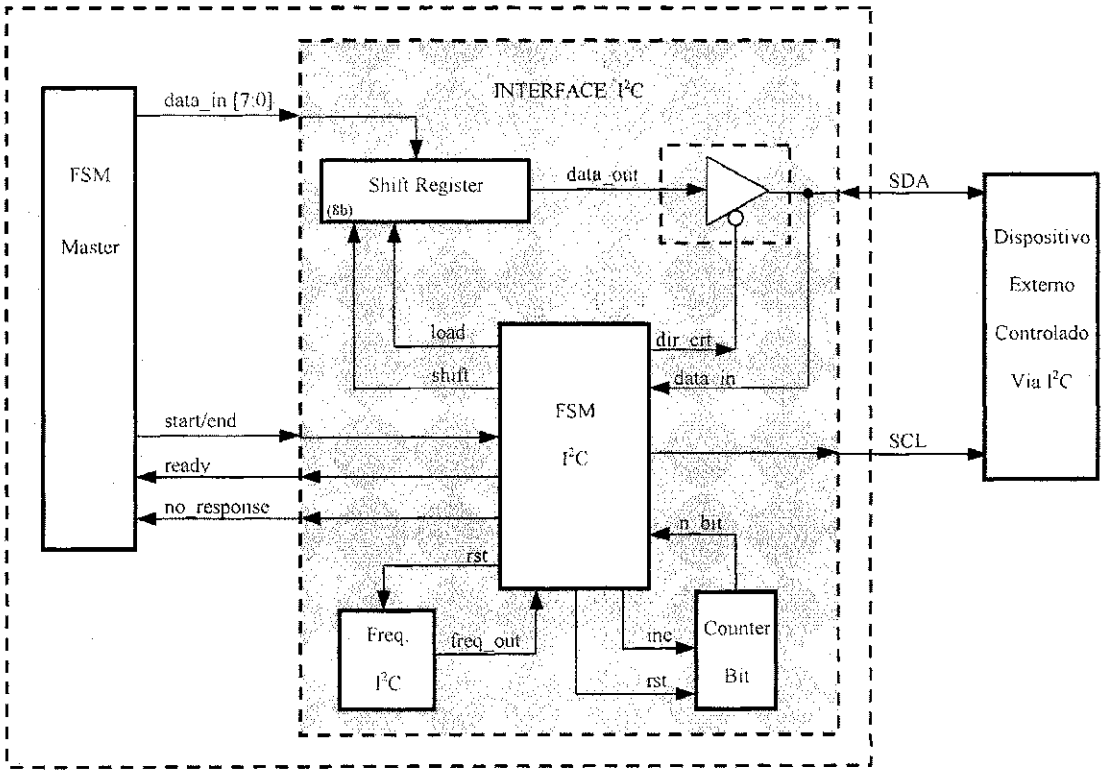


Figura 4-4: Diagrama em blocos da Interface I²C [7].

4.2.2.3 Interface com o Decodificador de Vídeo

O diagrama em blocos da Interface com o Decodificador de Vídeo é mostrado na Figura 4-5. A Interface com o Decodificador de Vídeo é o bloco responsável por capturar os elementos de imagem digitalizados pelo decodifcador de vídeo nas taxas especificadas para o projeto, quais sejam: 240 *pixel* por linha e 160 linhas por quadro. O princípio de funcionamento da interface com o decodificador de vídeo é o seguinte:

Inicialmente, a Interface com o Decodificador de Vídeo detecta o início de um campo ímpar através dos sinais de sincronismo vertical (V_signal) e de identificação de campo (Field_ident). Ao identificar o campo ímpar, aguarda o início da primeira linha (H_signal). Quando isto é detectado, aguarda um *pixel* válido na subida do clock de *pixel* (Clk_pixel). O *pixel* é armazenado em um dos *Buffers* de vídeo interno. O decodificador de vídeo, disponibiliza 720 *pixels* por linha. No entanto, as especificações do projeto requerem apenas 240 *pixels* por linha. Assim sendo, os dois *pixels* seguintes não serão armazenados, dando início a um processo cíclico de captura de um *pixel* e desprezo de dois subseqüentes. Quando o *Buffer* estiver com 240 *pixels*, sinaliza-se através do sinal *line_start* que há uma linha completa disponível para leitura. Para atender aos requisitos de números de linhas (160 linhas/quadro), a interface com o decodificador de vídeo processa uma linha e despreza as duas linhas seguintes, procedendo desta maneira até que um quadro completo tenha sido capturado. Quando todo o campo ímpar tiver sido analisado, a interface detecta o início do campo par e da primeira linha. No entanto, a primeira linha do campo par não é armazenada em função do entrelaçamento dos campos durante a apresentação da imagem final. O processamento do campo par inicia na segunda linha e, a partir deste ponto procede-se como no campo ímpar em relação aos *pixels* e linhas analisadas. Quando um quadro completo tiver sido capturado, a Interface com o Decodificador de Vídeo sinaliza (end_frame = 1). A máquina de estados que controla a leitura dos *pixels* também sinaliza o instante em que todos os *pixels* tiverem sido lido do *buffer* de vídeo (end_line).

A principal diferença entre esta interface e a proposta originalmente [7] é a inclusão de um outro *Buffer* de vídeo para armazenamento dos pixels capturados. Desta maneira, os *Buffers* estarão somente em escrita ou leitura, evitando a necessidade de um controle para verificação do instante em que o *buffer* está ocupado. As células da FPGA que são utilizadas para implementação destes *buffers* sempre estão disponíveis, pois são específicas para esta função. Na Figura 4-6 mostra-se o diagrama em blocos do *Buffer* de Vídeo.

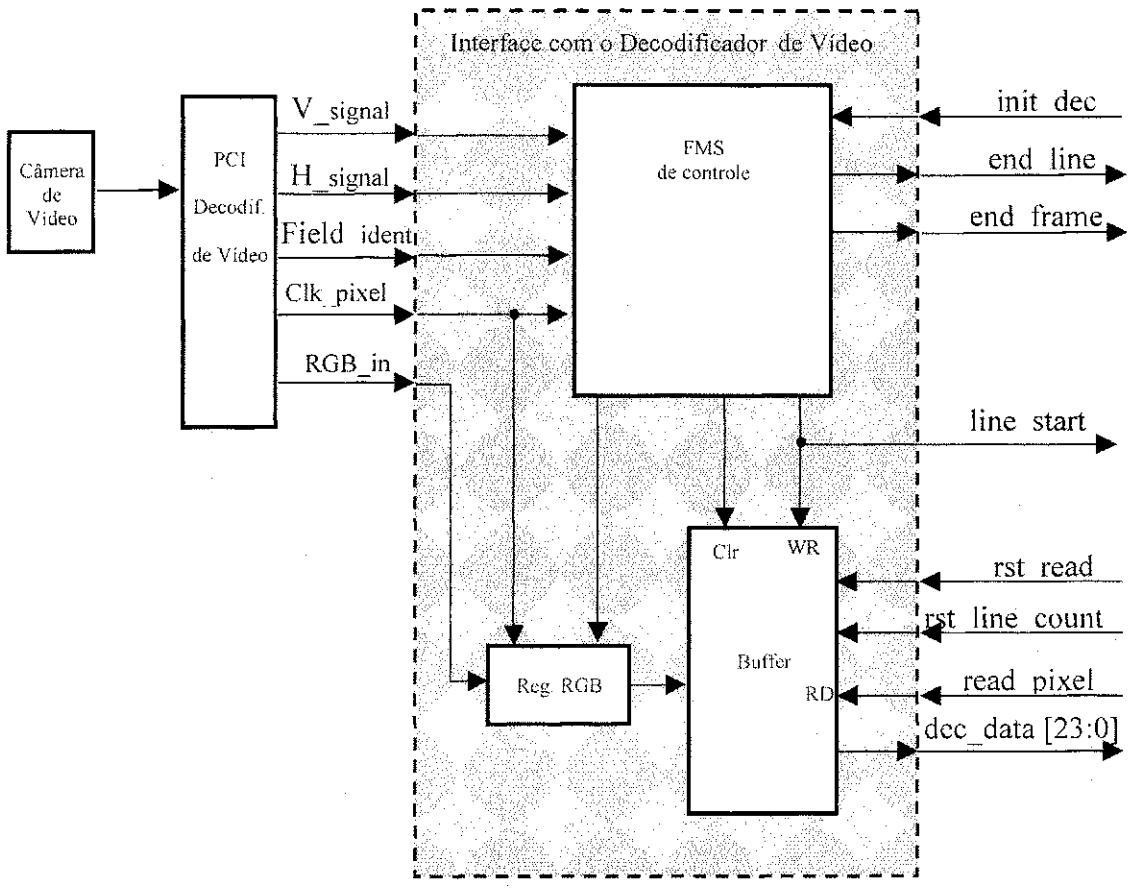


Figura 4-5: Diagrama em blocos da Interface com o Decodificador de Vídeo

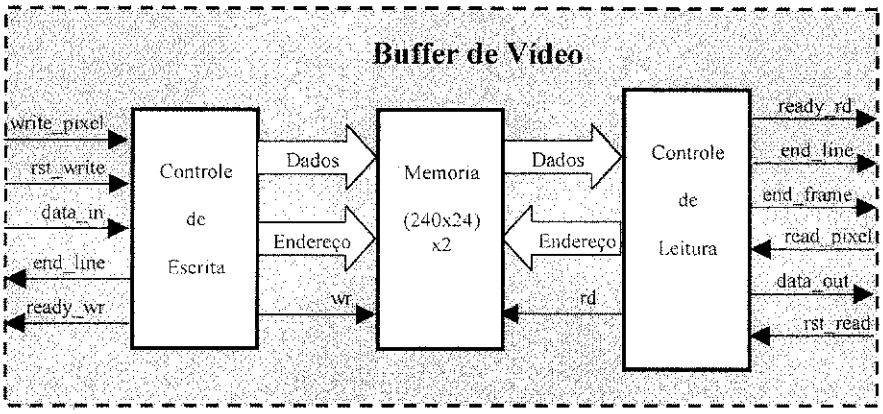


Figura 4-6: Diagrama em blocos do *Buffer* de Vídeo.

4.2.2.4 Multiplexador dos Sinais de Controle

A função do Multiplexador de Sinais de Controle é selecionar o bloco que comandará a Interface de Acesso a Memória da FPGA 01. Há 03 blocos que necessitam ler/escrever dados na memória, quais sejam: Inicialização da Memória, o Gerador de Parâmetros $E_i\_S_i$ e o Gerador de Parâmetros $\hat{\alpha}_i\_a_i$. A seleção dos sinais de controle é enviado pela Fsm_Principal. Na Figura 4-7 é mostrado o diagrama em blocos do Multiplexador de Sinais de Controle.

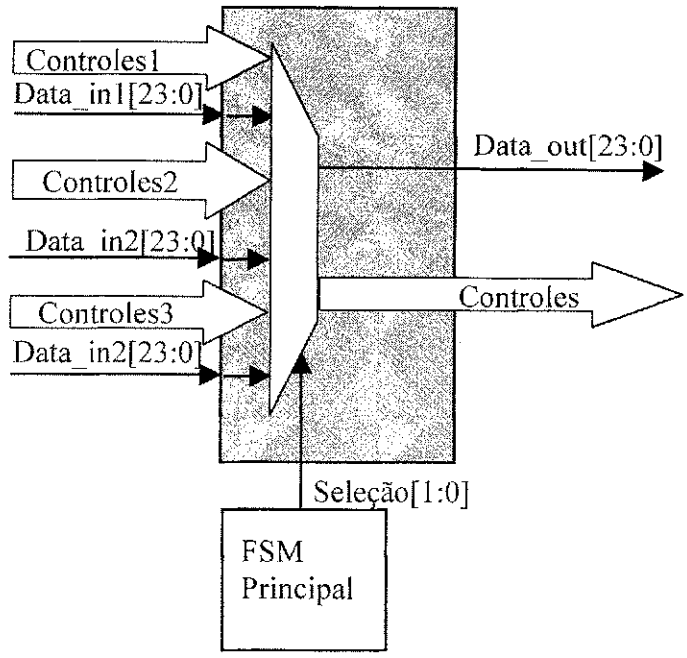


Figura 4-7: Diagrama em blocos do Multiplexador de Sinais de controle

4.2.2.5 Interface de Acesso à Memória

Na Figura 4-8 mostra-se o diagrama em blocos da Interface de Acesso à Memória. A Interface de acesso à memória é o bloco responsável pela escrita/leitura de dados na memória externa. Há um decodificador de endereços interno que, de acordo com nível lógico dos bits 17 e 16 do endereço, gera o sinal para habilitar o banco de memória externo. O endereço pode ser fornecido por um bloco externo ou gerado internamente. O bloco que estiver controlando a Interface de Acesso à Memória deverá informar através do sinal add_in_ext se o endereço é interno (add_int_ext = 0) ou externo (add_in_ext = 1). Quando se utiliza o gerador de endereços interno há ainda outras configurações quanto ao endereço inicial para escrita (rst_ctrl_wr), endereço inicial para leitura (rst_ctrl_rd), se o incremento do endereço de escrita será um ou dois (add_ctrl_wr) e se o incremento do endereço de leitura será um ou dois (add_ctrl_rd). Estas configurações são utilizadas pelos blocos Gerador de Parâmetros $E_i\_S_i$, Gerador de Parâmetros $\hat{\alpha}_i\_a_i$ e Gerador de Parâmetros

$C\hat{D}_i\_b_i$ para organizar os parâmetros calculados na fase de aprendizado na memória. A inclusão de geradores de endereços na Interface de acesso à memória evita a necessidade de geradores de endereços nos demais blocos. Além disso, há também um sinal (frame_end_mem) que informa o instante em que um quadro completo é armazenado na memória (240X160 = 38400 pixels).

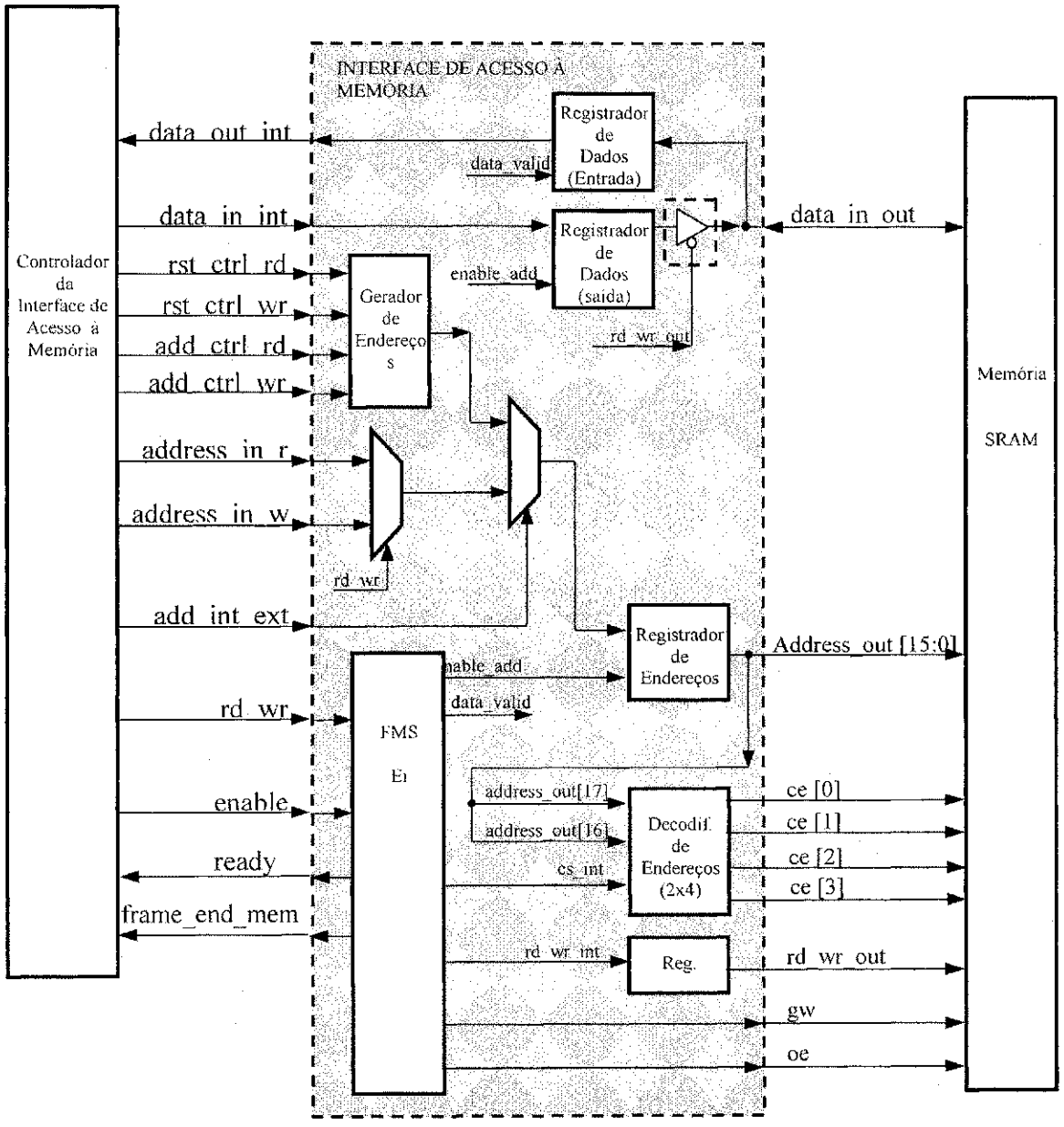


Figura 4-8: Diagrama em blocos da Interface de Acesso a Memória.

4.2.2.6 Inicialização da Memória

O módulo de Inicialização da Memória é responsável por preencher com zero (0) as posições de memória que serão utilizadas para o acúmulo temporário e definitivo dos parâmetros E_i, a_i e b_i. Isto é necessário, pois o acúmulo da média dos *pixels* e das variações médias das distorções de brilho e de cor são efetuados somando-se o valor acumulado na memória com o valor corrente processado e, desta maneira, o valor inicial deve ser zero (0).

4.2.2.7 *Buffer* de Parâmetros FPGA01

O diagrama em blocos do módulo *Buffer* de Parâmetros é mostrado na Figura 4-9.

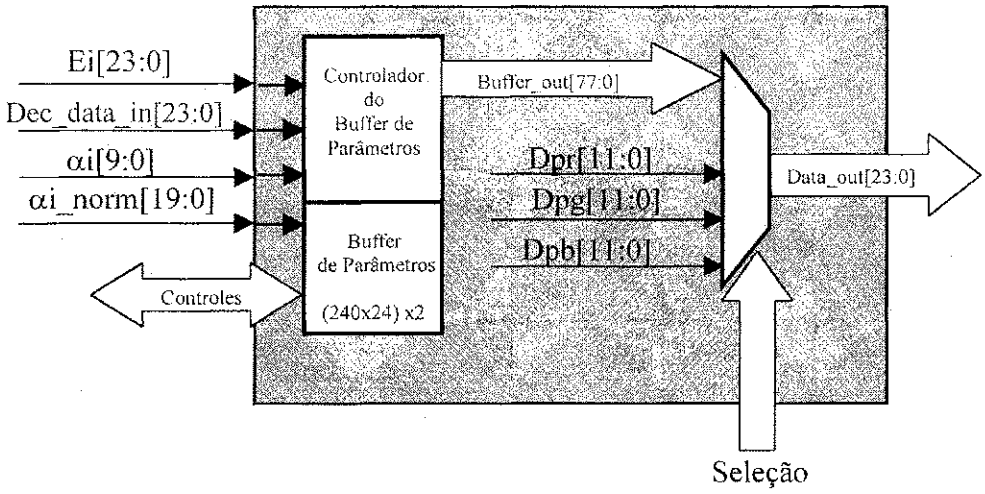


Figura 4-9: Diagrama em blocos do *Buffer* de Parâmetros – FPGA01

O *Buffer* de Parâmetros FPGA01 é composto por dois blocos principais, quais sejam: Controlador de *Buffer* de Parâmetros e Multiplexador de Sinais. O Controlador de *Buffer* de parâmetros contém dois *Buffers* com 240 posições de 78 bits utilizados para

armazenar os seguintes parâmetros: E_i, dec_data_in (pixel corrente), α_i e $\hat{\alpha}_i$. O *Buffer* de Parâmetros FPGA01 é habilitado cada vez que o parâmetro α_i é processado e os parâmetros do *pixel* corrente são armazenados no *Buffer* utilizado para escrita. Quando todas as 240 posições estiverem preenchidas, significa que os *pixels* de uma linha completa foram processados. O *Buffer* de Parâmetros FPGA01 sinaliza para o módulo denominado Transferência de Parâmetros na FPGA 02, através do sinal *line_start_buffer*, que há um *buffer* com uma linha completa processada disponível para transferência dos parâmetros para a FPGA 02. O *Buffer* de Parâmetros FPGA01 disponibiliza o *buffer* com os parâmetros correntes e utiliza o outro *buffer* para a escrita dos parâmetros dos *pixels* da próxima linha. Quando o módulo de Transferência de Parâmetros, localizado na FPGA 02, recebe a sinalização (*line_start_buffer*) começa a transferir os parâmetros armazenados no *buffer* bem como o $Dp_{(R)}$, $Dp_{(G)}$ e $Dp_{(B)}$ para o Registrador de Parâmetros da FPGA 02. Para transferir os parâmetros são gerados os sinais de controle para o *buffer* com os parâmetros a serem transferidos e a seleção do parâmetro a ser transferido é feita no Multiplexador localizado no *buffer* de Parâmetros FPGA01 que é controlado pelo módulo Transferência de Parâmetros.

4.2.2.8 Gerador de Parâmetros $E_i\_S_i$

O bloco Gerador de Parâmetros $E_i\_S_i$ é responsável pela obtenção da média e desvio padrão dos elementos de imagem. Quando a operação é de cálculo da média, a Fsm_Principal habilita o módulo de cálculo do parâmetro E_i e aguarda a sinalização de pronto. O módulo de cálculo do parâmetro E_i, ilustrado na Figura 4-10, soma os *pixels* armazenados no *Buffer* de Vídeo da Interface Decodificador de Vídeo com os *pixels* armazenados na memória externa e acumula-os na memória externa até que um quadro completo seja processado. Ao finalizar o processamento de um quadro completo, o módulo de cálculo do parâmetro E_i sinaliza para a Fsm_Principal. A Fsm_Principal habilita novamente o módulo de cálculo do parâmetro E_i e o ciclo se repete até que 64 quadros tenham sido processados.

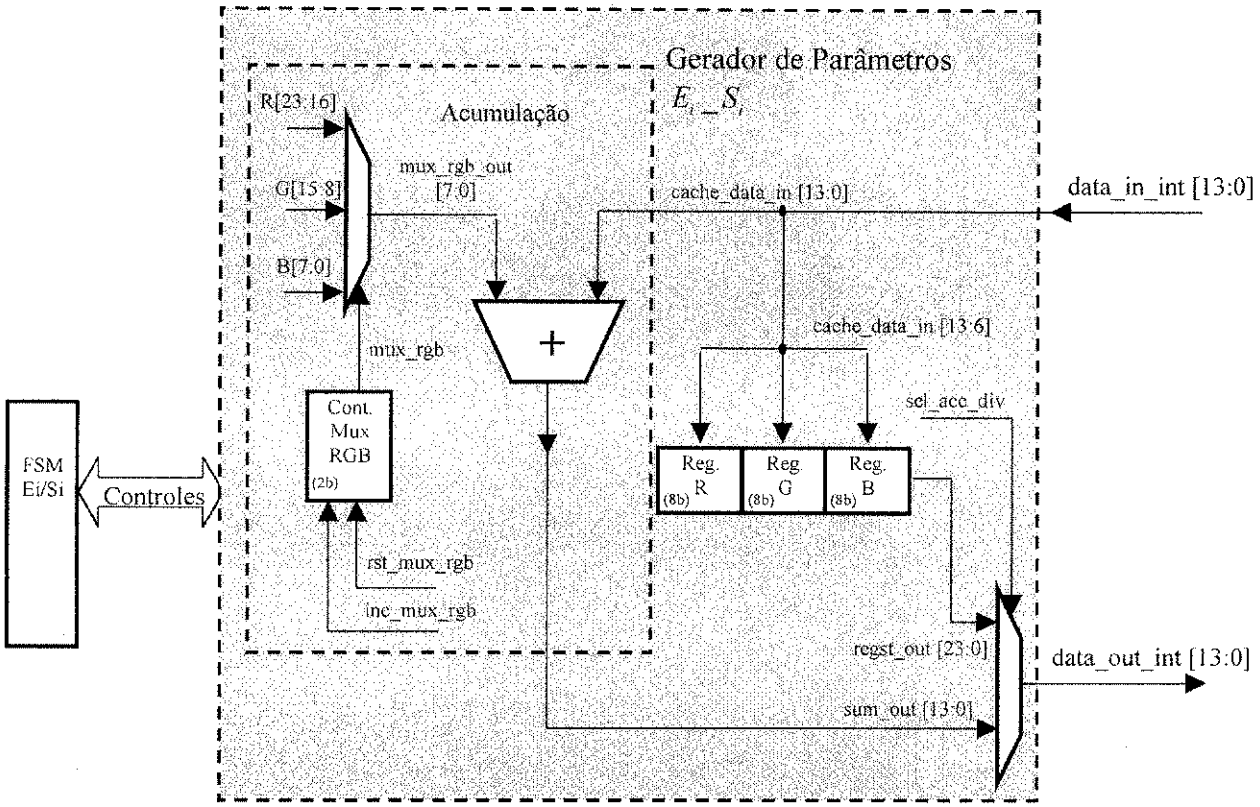


Figura 4-10: Diagrama em blocos do módulo de cálculo do parâmetro E_i.

Ao final dos 64 quadros, a Fsm_Principal habilita o módulo de cálculo do parâmetro E_i para calcular a média dos *pixels* acumulados na memória externa. O módulo de cálculo do parâmetro E_i procede com o cálculo da média e armazena estes valores na memória externa.

Quando o cálculo é desvio padrão (S_i), a Fsm_Principal habilita o módulo de cálculo do parâmetro S_i e aguarda a resposta. O módulo de cálculo do parâmetro S_i procede com o cálculo do desvio padrão calculando a diferença entre o pixel corrente e o

seu respectivo valor médio, em seguida calcula o valor da diferença ao quadrado e soma com o valor previamente calculado. Neste caso, o acúmulo é efetuado em registradores internos. Quando um quadro completo é processado, o sinal de ready_ei informa para a Fsm_Principal o término do processo. A Fsm_Principal, habilita novamente o módulo de cálculo do parâmetro S_i e o ciclo se repete até que 27 quadros tenham sido processados. Ao final desta etapa, a Fsm_Principal habilita o módulo de cálculo do parâmetro S_i para efetuar o cálculo final do desvio padrão. Os valores acumulados são divididos por 4096 para obtenção dos valores finais de $Dp_{(R)}$, $Dp_{(G)}$ e $Dp_{(B)}$. Novamente, o módulo de cálculo do parâmetro S_i sinaliza para a Fsm_Principal o término deste cálculo. O diagrama em blocos do módulo de processamento do S_i é mostrado na Figura 4-11.

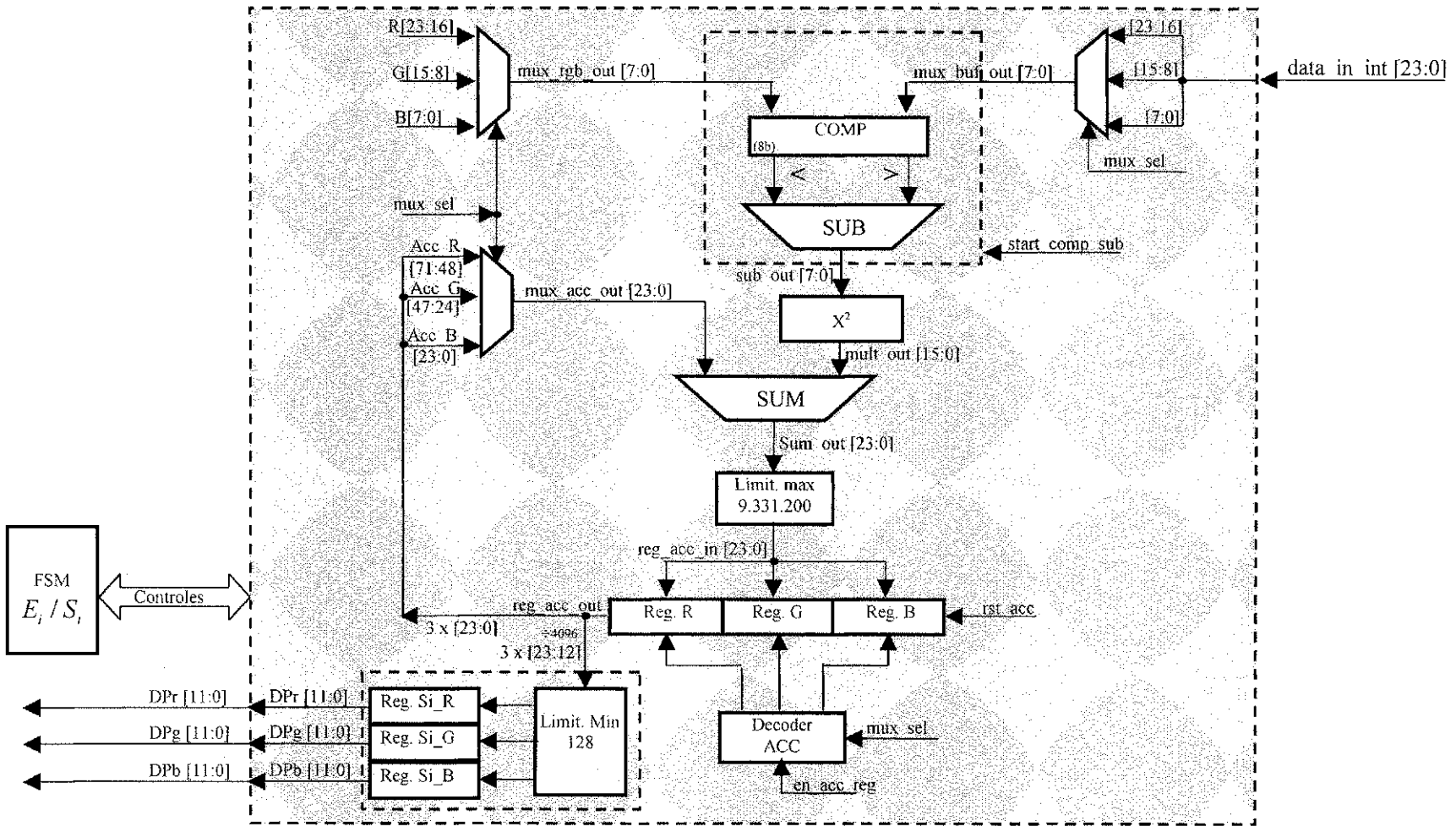


Figura 4-11: Diagrama em blocos do módulo de cálculo do parâmetro S_i.

4.2.2.9 Gerador de Parâmetros $\hat{\alpha}_i\_a_i$

Na Figura 4-12 mostra-se o diagrama em blocos do Gerador de Parâmetros $\hat{\alpha}_i\_a_i$.

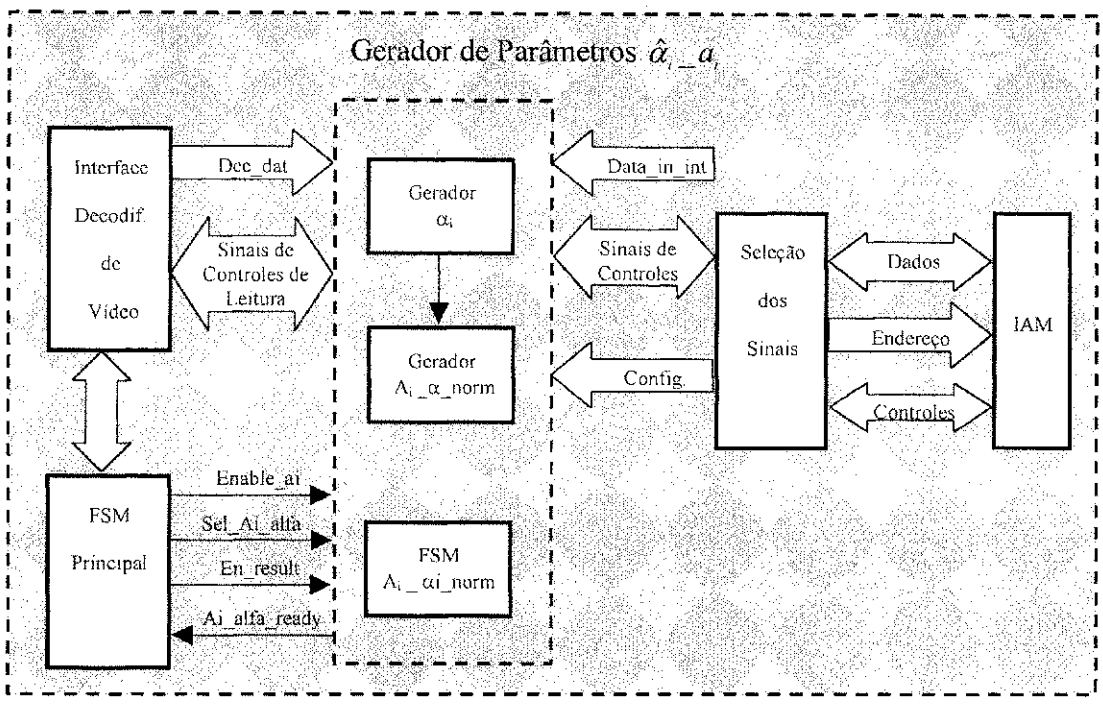


Figura 4-12: Diagrama em blocos do Gerador de Parâmetros $\hat{\alpha}_i\_a_i$.

O módulo Gerador de Parâmetros $\hat{\alpha}_i\_a_i$ é responsável pelo cálculo dos parâmetros $\hat{\alpha}_i$ e a_i dos elementos de imagem. Para calcular o parâmetro a_i o Gerador de Parâmetros $\hat{\alpha}_i\_a_i$ processa os *pixels* armazenados no *Buffer* de Vídeo da Interface Decodificador de Vídeo e acumula os resultados parciais na memória externa até que um quadro completo seja processado. Para obtenção do valor do parâmetro a_i, calcula-se o valor da distorção de brilho segundo o diagrama em blocos ilustrado na Figura 4-13. Ao finalizar o processamento de um quadro completo, o Gerador de Parâmetros $\hat{\alpha}_i\_a_i$ sinaliza para a Fsm_Principal. A Fsm_Principal habilita novamente o Gerador de Parâmetros $\hat{\alpha}_i\_a_i$ e o ciclo se repete até que 64 quadros tenham sido processados. Ao final dos 64 quadros, a

Fsm_Principal habilita o Gerador de Parâmetros $\hat{\alpha}_i\_a_i$, através dos sinais enable_ai e en_result, para calcular os valores finais do parâmetro a_i e armazená-los na memória externa.

Para calcular a distorção de brilho normalizada ($\hat{\alpha}_i$) o Gerador de Parâmetros $\hat{\alpha}_i\_a_i$ calcula primeiramente a distorção de brilho com duas casas decimais e em seguida divide o resultado pelo parâmetro a_i previamente calculado e armazenado na memória externa.

O módulo de processamento do parâmetro α_i foi implementado visando a otimização em área e velocidade. Para tal, os divisores do numerador e denominador da expressão de α_i foram substituídos por multiplicadores. Nas tabelas 4-3 e 4-4 apresenta-se o número de elementos lógicos e a quantidade de ciclos de operação necessários para a implementação da expressão de α_i utilizando divisores e multiplicadores, respectivamente. Conforme mostrado na Tabela 4-4 a implementação com multiplicadores reduz em 1% a quantidade de elementos lógicos e o número de ciclos de operação reduz de 70 para 48 ciclos e isto é fundamental na implementação do sistema de extração de objetos em tempo real.

Tabela 4-3: Número de Elementos Lógicos e Ciclos de *clock* utilizando divisores no numerador e denominador da Expressão αi.

Gerador de Parâmetro αi – Divisores no numerador e denominador da expressão de α_i				
Bloco	N.o Elementos Lógicos (%)	Quantidade	N.o ciclos Operação	Soma parcial Elementos Lógicos (%)
Multiplicador 8x8	1	3	2	3
Divisor 31/12	9	3	24 para ($2^{31}/2^{7}$)	27
Divisor 33/26	14	1	26 para ($2^{33}/2^{7}$)	18
Soma Total N.o Ciclos de Operação			70 ciclos de clock	
Soma Total de Elementos Lógicos				44 %

Tabela 4-4: Número de Elementos Lógicos e Ciclos de *clock* utilizando multiplicadores no numerador e denominador da Expressão αi.

Gerador de Parâmetro αi – Multiplicadores no numerador e denominador da expressão de αi				
Bloco	N.o Elementos Lógicos (%)	Quantidade	N.o ciclos Operação	Soma parcial Elementos Lógicos (%)
Multiplicador 8x8	1	3	2	3
Multiplicador 12x12	2	3	2	6
Multiplicador 24x16	6	3	2	18
Divisor 49/42	16	1	42 para $(2^{49}/2^{7})$	16
Soma Total N.o Ciclos de Operação			48 ciclos de clock	
Soma Total de Elementos Lógicos				43 %

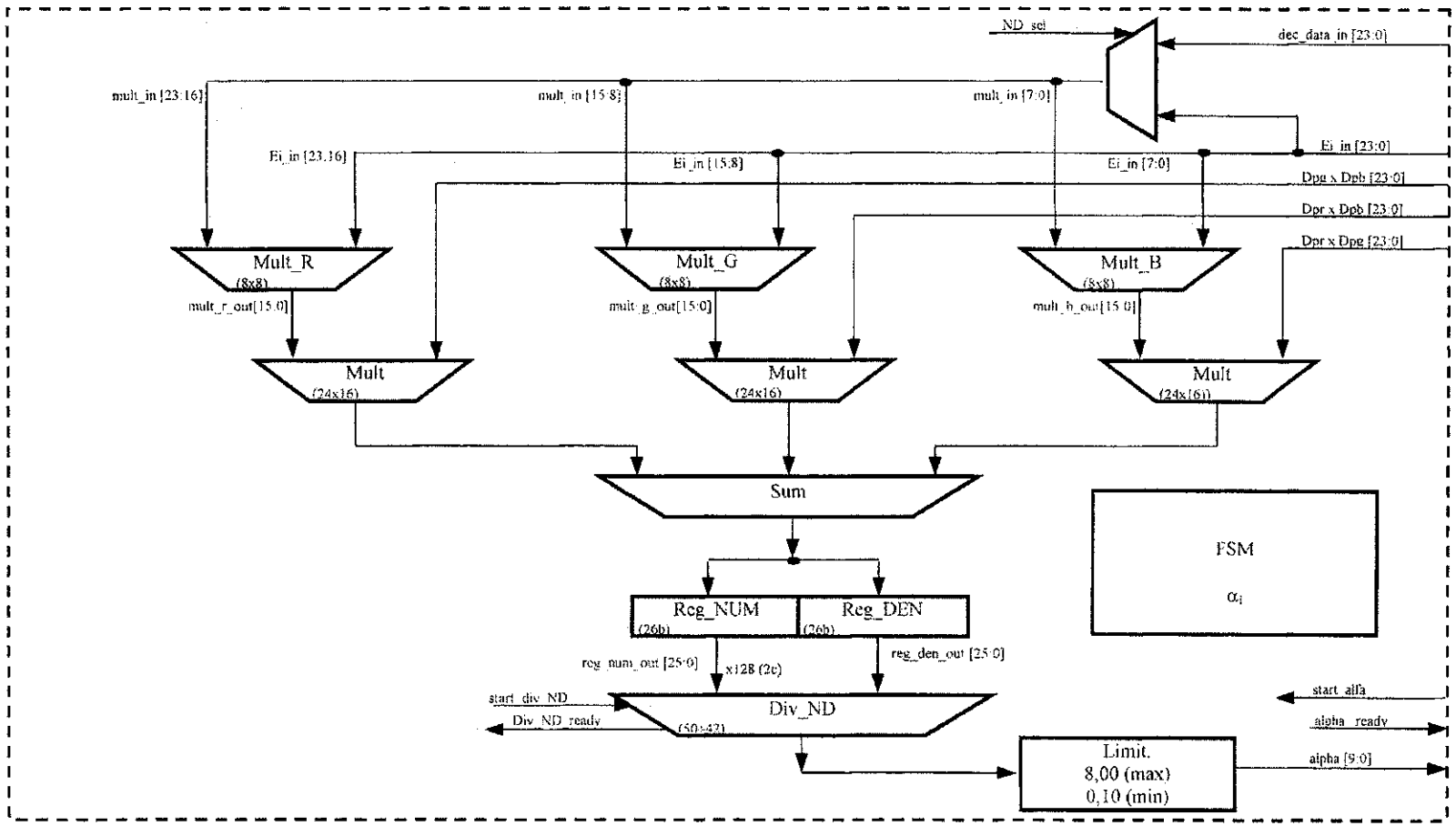


Figura 4-13: Diagrama em blocos do módulo de cálculo do parâmetro αi.

4.2.2.10. Controlador da Interface de Acesso à Memória

Na Figura 4-14 mostra-se o diagrama em blocos do Controlador da Interface de Acesso à Memória. O Controlador da Interface de Acesso à Memória é responsável por selecionar o bloco que controla a Interface de Acesso à Memória (IAM) de acordo com a requisição efetuada através das entradas int0, int1 e int2. Quando se pretende escrever ou ler um dado da memória, deve-se fazer a requisição ao Controlador da Interface de Acesso à Memória mantendo o sinal de interrupção (int) em alto. O Controlador da Interface de Acesso à Memória aguardará a finalização de leitura ou escrita corrente, ao final sinaliza para o bloco requisitante através do sinal ack que a Interface de Acesso à Memória está disponível para escrita/leitura. Este controle é necessário, pois temos blocos distintos lendo e escrevendo dados na memória. O Gerador de Parâmetros $C\hat{D}_{i\_}b_i$ lê os valores do parâmetro b_i para calcular a distorção de cor normalizada ($C\hat{D}_i$), o bloco de segmentação atualiza constantemente a área da memória destinada ao quadro a ser exibido pelo codificador de vídeo e o bloco de apresentação lê o quadro a ser exibido pelo codificador também da memória. Portanto, o Controlador da Interface de Acesso à Memória organiza o acesso a IAM, evitando conflitos, priorizando a leitura do parâmetro b_i, em seguida a atualização do quadro e finalmente a leitura do quadro a ser exibido.

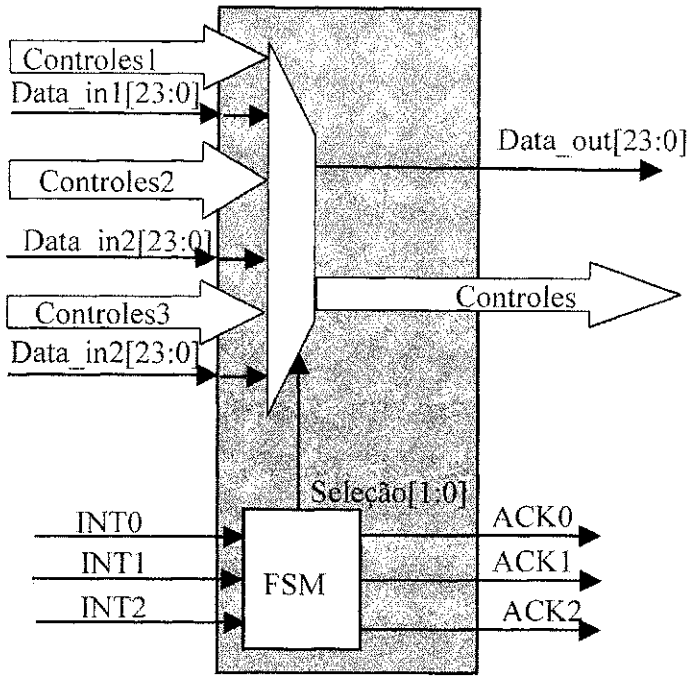


Figura 4-14: Diagrama em blocos do Controlador da Interface de Acesso à Memória (IAM).

4.2.2.11. Registrador de Parâmetros

Na Figura 4-15 mostra-se o diagrama em blocos do Registrador de Parâmetros. O Registrador de Parâmetros é composto por 07 registradores utilizados para registrar os parâmetros transferidos da FPGA 01, quais sejam: $Dp_{(R)}$, $Dp_{(G)}$, $Dp_{(B)}$, $\hat{\alpha}_i$, α_i, E_i, Dec_data (pixel corrente). A seleção dos registradores é feita através do sinal (sel [2:0]) que é decodificado internamente para habilitar somente um registrador em cada ciclo de escrita. O sinal de escrita (load_data) e de seleção (sel_data_mem) são enviados pelo bloco de Transferência de parâmetros. Quando o dado é registrado, o bloco Registrador de Parâmetros sinaliza através do sinal load_ready.

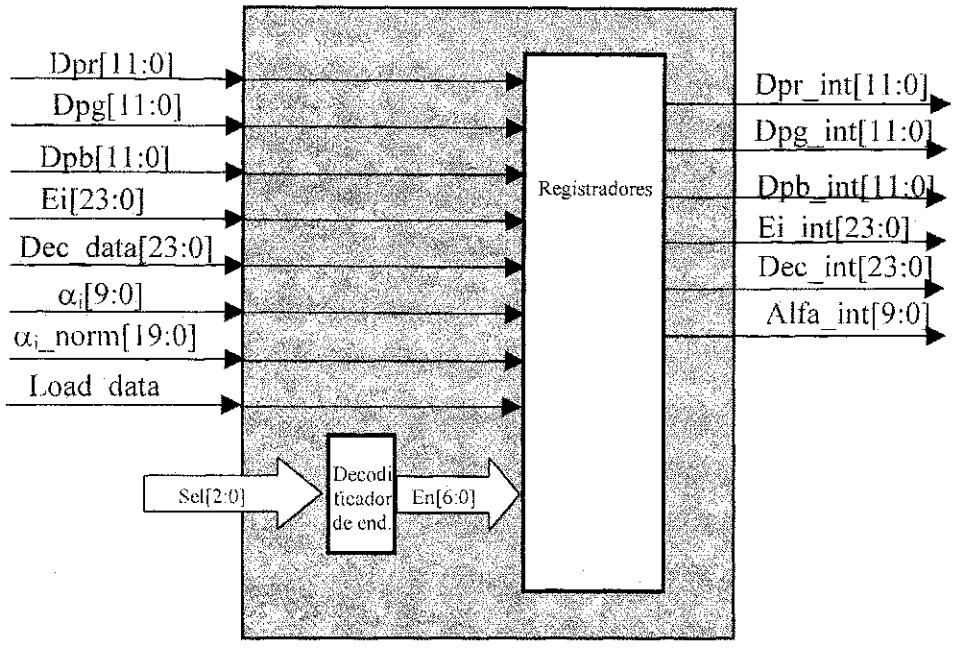


Figura 4-15: Diagrama em blocos do Registrador de Parâmetros.

4.2.2.12.. *Buffer* de Parâmetros FPGA02

Na Figura 4-16 mostra-se o diagrama em blocos do *Buffer* de Parâmetros FPGA02. O *Buffer* de Parâmetros FPGA02 é responsável por armazenar os seguintes parâmetros: $\hat{\alpha}_i$, Dec_data (pixel corrente) e sum_out que é o resultado parcial do Gerador de Parâmetro CD-I. O controle de escrita é efetuado pelo módulo de Transferência de Parâmetros e o controle de leitura pelo módulo Gerador de Parâmetro CD-II.

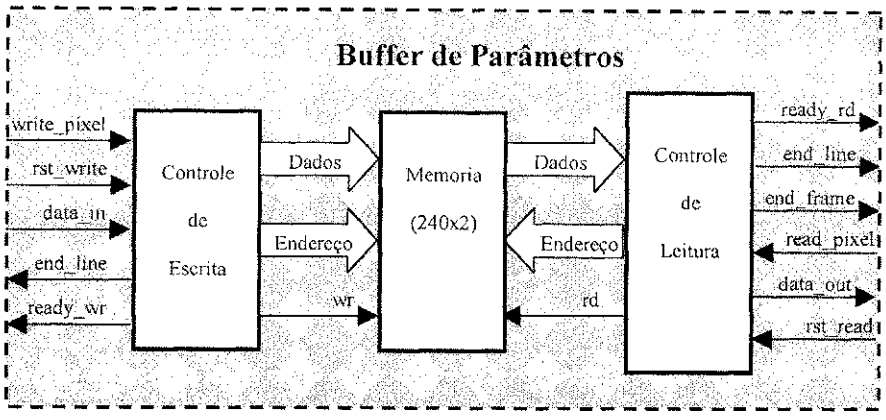


Figura 4-16: Diagrama em Blocos do *Buffer* de Parâmetros FPGA02.

4.2.2.13.. Gerador de Histograma e Limiar

O Gerador de Histograma e Limiar é composto basicamente por um *Buffer* de 2048 posições, um somador e uma máquina de estados. O *buffer* é utilizado para registrar o número de ocorrências do parâmetro $C\hat{D}_i$. Cada posição do *buffer* representa o valor do $C\hat{D}_i$ cuja faixa de variação é de 0 a 2047. Assim sendo, durante a construção do histograma, para cada valor de $C\hat{D}_i$, lê-se a posição de memória correspondente e incrementa-se o valor contido nesta posição de memória. Uma vez finalizado o registro das ocorrências para o espaço amostral determinado (64 quadros), a Fsm_Principal habilita a obtenção do limiar. O valor admissível para erros de detecção foi estabelecido em 0,01%, o que, em um universo de 240x160x32=1.228.800 amostras, corresponde a 1228 ocorrências. Para obtenção do limiar, todas as ocorrências são lidas e acumuladas, decrementando-se o endereço desde a menor probabilidade (o valor 2047), até que tenhamos acumulado um valor superior a 1228. Quando este valor é acumulado, a última posição do *buffer* analisada será o valor do limiar utilizado para classificar os elementos de imagem.

4.2.2.14.. Segmentação

O diagrama em blocos do módulo Segmentação é mostrado na Figura 4-17. A Segmentação é o bloco responsável por classificar o *pixel* baseando-se nos valores de $C\hat{D}_i$, $\hat{\alpha}_i$ e do limiar de comparação. Além de classificar o *pixel*, também organiza na memória externa os quadros a serem exibidos pelo codificador de vídeo externo. A outra função do bloco de segmentação é organizar na memória externa os valores da variação média da distorção de cor (b_i) durante a fase de aprendizado.

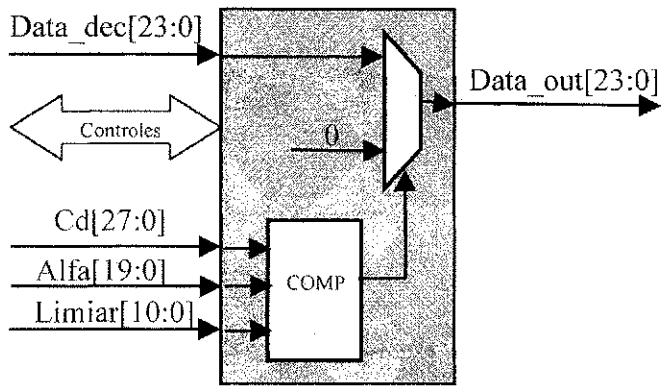


Figura 4-17: Diagrama em blocos do módulo de Segmentação.

4.2.2.15 Gerador dos Parâmetros $C\hat{D}_i\_b_i$

O Gerador de Parâmetros $C\hat{D}_i\_b_i$ é o bloco responsável por calcular os parâmetros b_i e $C\hat{D}_i$. Em função do número de ciclos disponíveis para o processamento de cada *pixel* (54 ciclos), distribui-se o processamento dos parâmetros $C\hat{D}_i$ e b_i em dois blocos denominados Gerador de Parâmetros CD_i_I e Gerador de Parâmetros CD_i_II. No bloco denominado Gerador de Parâmetros CD_i_II, obtêm-se a soma das parcelas da equação 3-6 ainda sem a extração da raiz quadrada. O valor resultante da soma das parcelas juntamente

com o $\hat{\alpha}_i$ e Dec_data (*pixel* corrente) são armazenados no *Buffer* de Parâmetros da FPGA02. Quando o *Buffer* de Parâmetros contém 240 parâmetros disponíveis, o módulo Transferência de Parâmetros sinaliza para o Gerador de Parâmetros CD_i_I através do sinal *line_start*. Quando a operação é cálculo do parâmetro b_i, o Gerador de Parâmetros CD_i_I começa a ler os parâmetros armazenados no *Buffer* de Parâmetros da FPGA02 para calcular a distorção de cor e, posteriormente, somar ao resultado desta operação os valores do parâmetro b_i acumulados na memória externa. O resultado final é enviado para o bloco de segmentação que o organiza na memória externa. Para obtenção da distorção de cor normalizada ($C\hat{D}_i$), o valor do CD_i com duas casas decimais é dividido pelo valor do parâmetro b_i armazenado na memória externa. Os diagramas em blocos do Gerador de Parâmetros CD_i_I e do Gerador de Parâmetros CD_i_II estão ilustrados nas Figura 4-18 e Figura 4-19, respectivamente.

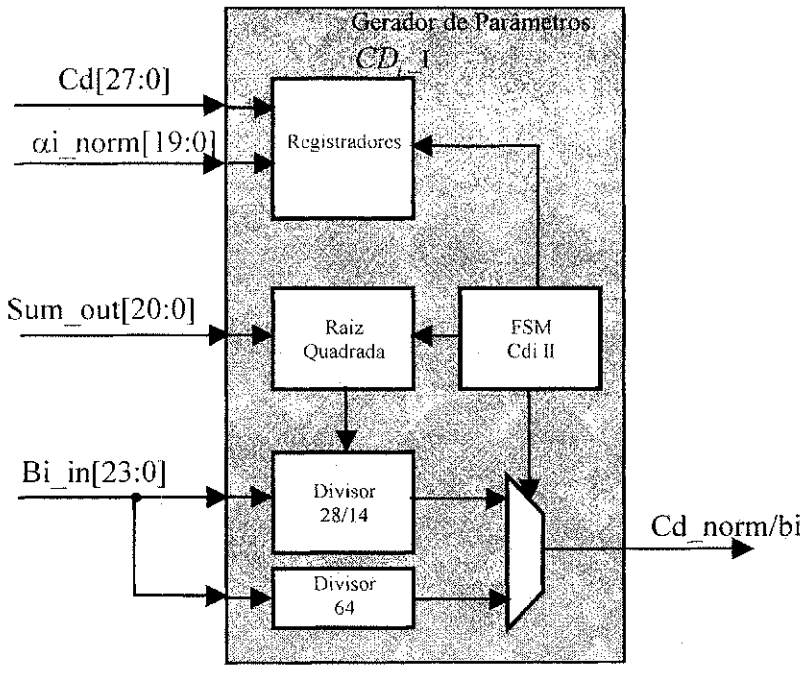


Figura 4-18: Diagrama em blocos do Gerador de Parâmetro CD_i_I.

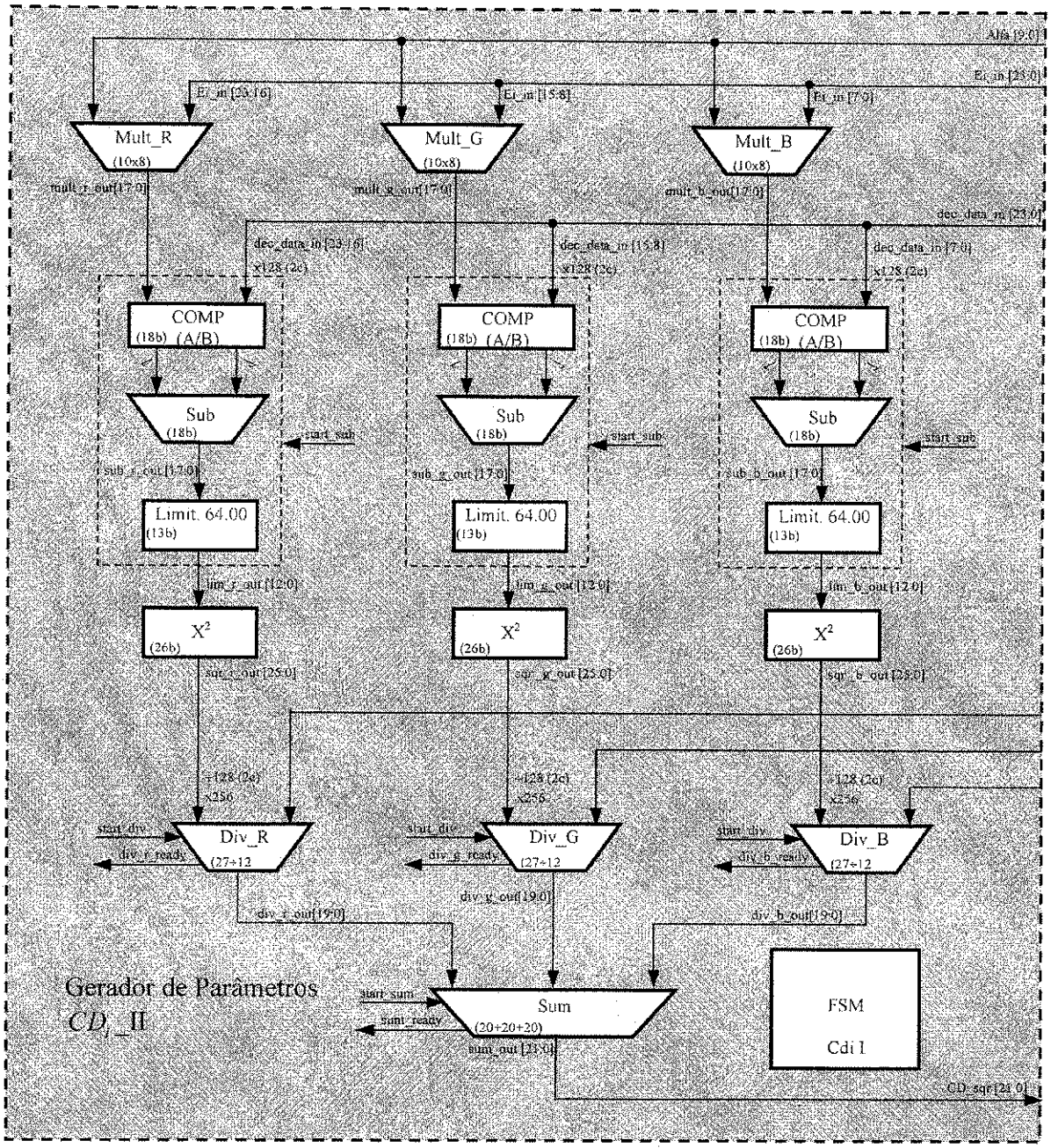


Figura 4-19: Diagrama em Blocos do Gerador de Parâmetro CD_i_II

4.2.2.16.. Apresentação do Objeto

O diagrama em blocos do módulo denominado Apresentação do Objeto é mostrado na Figura 4-20. O módulo denominado Apresentação do Objeto é responsável por efetuar a leitura dos *pixels* obtidos do processo de classificação que se encontram em uma área específica da memória. Basicamente, o módulo denominado Apresentação do Objeto é composto por quatro *buffers* de 240 posições destinados ao armazenamento de duas linhas pares e duas linhas ímpares. Quando o módulo é habilitado, as linhas ímpares e as linhas pares do quadro 01 são lidas e armazenadas nos *buffers* internos. Sempre há a leitura de uma linha ímpar e uma linha par até que todo o quadro tenha sido lido. Ao término da leitura do quadro 01, o bloco Apresentação de Objeto inicia a leitura do quadro 02 da mesma maneira que no quadro 01 dando início a um ciclo constante de leitura das linhas ímpares e pares dos quadros 01 e 02. Enquanto um par de linhas é armazenado internamente, os pares de linhas previamente lidos ficam disponíveis para o bloco Interface Codificador de Vídeo enviá-los para o Codificador de Vídeo.

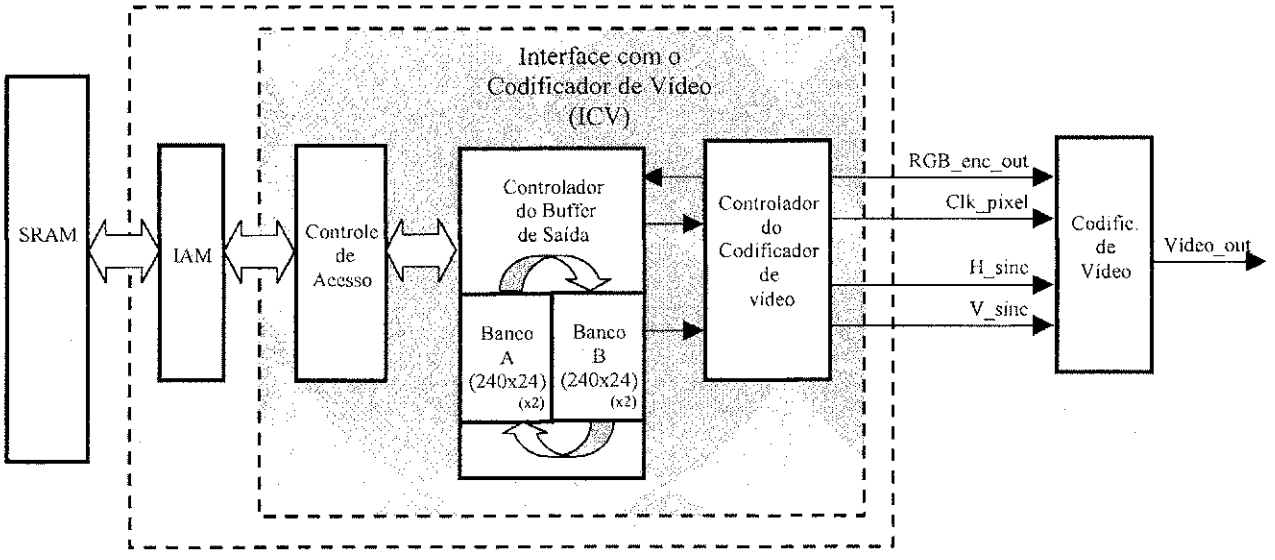


Figura 4-20: Diagrama em blocos do módulo Apresentação do Objeto

4.3 Modelagem em HDL

Na fase de Modelagem em HDL, cada bloco do sistema foi implementado utilizando a linguagem de descrição de hardware Verilog [30] a nível comportamental RTL (*Register Transfer Level*).

4.4 Síntese

Uma vez validado a descrição comportamental de cada bloco, fez-se a síntese de cada modelo que consiste em converter a descrição comportamental para estruturas de hardware de uma tecnologia definida. Neste caso, a tecnologia é a FPGA da plataforma (APEX-EP20K200EQC240-2X). Portanto, o modelo comportamental foi convertido para uma descrição estrutural baseada nos elementos físicos primitivos deste dispositivo, isto é, os elementos lógicos.

Esta conversão foi feita por uma ferramenta de síntese. Utilizou-se o Leonardo Spectrum [31] para esta tarefa. Ao finalizar a síntese do modelo, a ferramenta gera um arquivo em formato EDIF[4], com a descrição do modelo em termos de elementos lógicos do FPGA.

4.5 Leiaute

Finalizada a síntese do modelo, utilizou-se o arquivo EDIF na ferramenta de leiaute específica do fabricante da FPGA, para posicionamento e interligação física automática da descrição estrutural gerada pela ferramenta de síntese. Além disso, a ferramenta de leiaute

[4] EDIF: Electronic Design Interchange Format. É um formato de arquivo padronizado industrialmente para transmissão de dados de projetos

também gera um arquivo com a descrição de atrasos extraídos do layout implementado na FPGA e os arquivos para configuração física da FPGA. Utilizou-se o Quartus [27] para esta tarefa.

4.6 Verificação Comportamental e Estrutural

Uma vez modelado o bloco em HDL (*Hardware Description Language*), fez-se a simulação do modelo no ModelSim [32] para validar-se a funcionalidade. Para isto, criou-se um *testbench*[5] que aplica estímulos ao modelo de cada bloco e analisa as respostas a tais estímulos e as compara com as respostas esperadas. Os estímulos dos sinais de entrada de cada bloco são gerados por modelos também descritos em HDL. A verificação comportamental consiste em validar a funcionalidade do modelo descrito em HDL sem considerar os atrasos de cada célula do dispositivo de lógica programável. Uma vez validado o modelo comportamental, foi realizada a verificação estrutural que consiste em validar o modelo com os atrasos reais do dispositivo de lógica programável. A seguir são apresentados os módulos de testes utilizados para validação de cada bloco do sistema.

4.6.1 Validação do *Buffer* de Vídeo

Para validar-se o *buffer* de vídeo, foi implementada uma máquina de estados que armazena dados conhecidos no *buffer* e, em seguida, efetua a leitura e comparação para averiguar se os dados foram corretamente armazenados. Se alguma posição do *buffer* estiver com um dado errado, a máquina de estados de teste sinaliza com um sinal sempre em nível alto (1). Se todos os dados estiverem corretos, um sinal de 0,5Hz é gerado em um diodo emissor de luz (LED) externo.

Uma vez validado este módulo, os demais blocos do sistema foram implementados e utilizou-se o *buffer* de vídeo para validação de cada bloco do sistema. Na Figura 4-21

[5] *Testbench*: Esquema de validação de circuitos digitais composto por gerador de estímulos, verificador de respostas e modelo do sistema a ser validado.

mostra-se o diagrama em blocos dos módulos utilizados para verificação do *Buffer* de Vídeo.

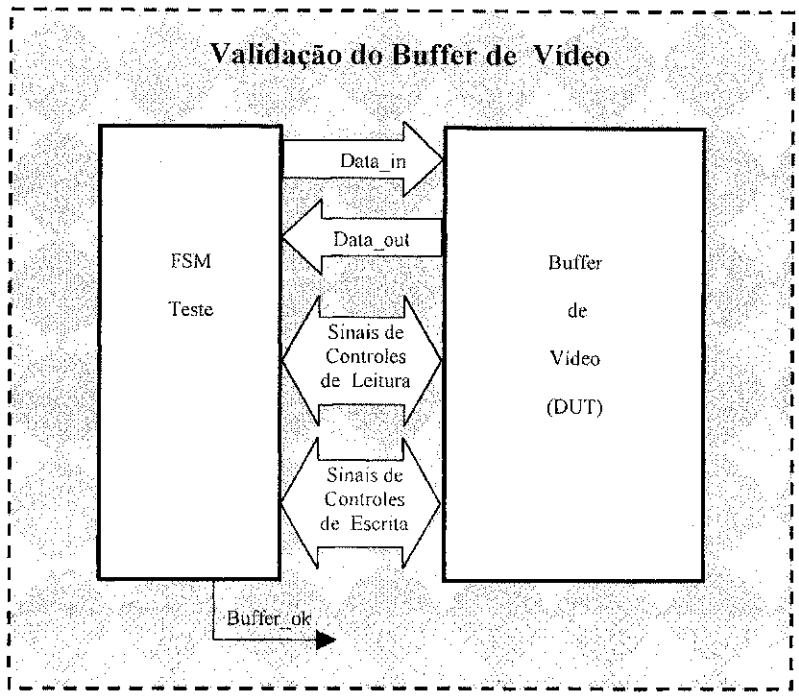


Figura 4-21: Diagrama em Blocos para Validação do *Buffer* de Vídeo.

4.6.2 Validação do Gerador de Parâmetro E_i

Na Figura 4-22 é mostrado o diagrama em blocos da plataforma de testes utilizada para validação do Gerador de Parâmetro E_i. Basicamente, há dois *buffers* de vídeo utilizados para armazenar os valores de teste, uma máquina de estados de teste (Fsm_teste), um multiplexador para seleção dos sinais de controle do *buffer* 2 e o Gerador de Parâmetro E_i. Para processar o E_i, são necessários os *pixels* do decodificador de vídeo e os valores armazenados na memória que inicialmente devem ser zeros. O *buffer* de vídeo 01 está "emulando" o decodificador de vídeo e o *buffer* 02 está "emulando" a memória externa.

Inicialmente, a máquina de estados de teste armazena alguns valores conhecidos no *buffer* 01 e inicializa com zero todas as posições do *buffer* 02. Em seguida, a máquina de estados de teste habilita o Gerador de Parâmetro E_i. Ao término do processamento, a máquina de estados de teste verifica se as médias calculadas e armazenadas no *buffer* 02 correspondem aos valores esperados. Se todos os valores estiverem corretos, um sinal de 0,5Hz é gerado em um LED externo. Do contrário, o LED ficará sempre aceso, indicando erro em um dos valores.

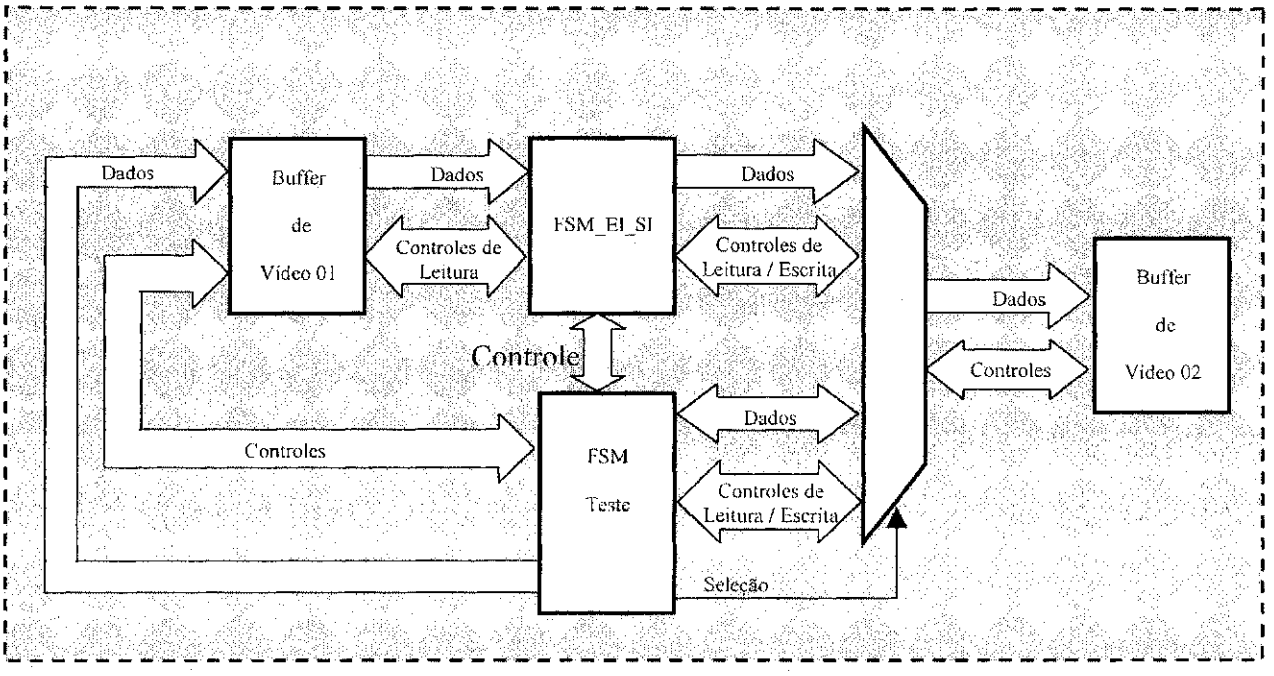


Figura 4-22: Diagrama em Blocos para Validação do Gerador de Parâmetro $E_i\_S_i$.

4.6.3 Validação do Gerador de Parâmetro S_i

A plataforma utilizada para validação do Gerador de Parâmetro S_i é a mesma utilizada para validação do Gerador de Parâmetro E_i. Neste caso, o *buffer* 01 é utilizado para "emular" os valores do decodificador de vídeo enquanto que o *buffer* 02 contém os valores das médias dos *pixels* necessários para o processamento do parâmetro S_i.

Inicialmente a máquina de estados de teste armazena valores predeterminados nos *buffers* 01 e 02. Em seguida, habilita o Gerador de Parâmetro S_i para proceder com o cálculo do desvio padrão. Ao término do processamento, a máquina de estados de teste verifica os valores dos parâmetros $Dp_{(R)}$, $Dp_{(G)}$ e $Dp_{(B)}$ na saída do Gerador de Parâmetro S_i. Se todos os valores estiverem corretos, um sinal de 0,5Hz é gerado em um LED externo. Caso contrário, o LED permanece aceso, indicando erro em um dos valores calculados.

4.6.4 Validação do Gerador de Parâmetro $\hat{\alpha}_i\_a_i$

Na Figura 4-23 mostra-se o diagrama em blocos da plataforma de testes utilizada para validação do Gerador de Parâmetros $\hat{\alpha}_i\_a_i$. O *buffer* 01 contém os valores dos pixels do decodificador de vídeo e o *buffer* 02 contém os valores médios dos *pixels* e os valores acumulados do parâmetro a_i. Além disso, também são necessários os valores do $Dp_{(R)}$, $Dp_{(G)}$ e $Dp_{(B)}$ para o cálculo dos parâmetros a_i e $\hat{\alpha}_i$. Os valores dos parâmetros $Dp_{(R)}$, $Dp_{(G)}$ e $Dp_{(B)}$ são fornecidos pela máquina de estados de teste durante a inicialização. Quando o processamento é do parâmetro $\hat{\alpha}_i$, a máquina de estados de teste habilita o Gerador de Parâmetros $\hat{\alpha}_i\_a_i$ para calcular os valores de $\hat{\alpha}_i$ e a medida que os valores são calculados, a máquina de estados de teste armazena-os em um *buffer* interno. Ao término do processamento, estes resultados são comparados com os valores esperados. Para validar a geração do parâmetro a_i, a máquina de estados de teste habilita o Gerador de Parâmetros $\hat{\alpha}_i\_a_i$ para processar os valores do parâmetro a_i e ao final do processamento, a máquina de estados de testes verifica se os valores do parâmetro a_i armazenados no *buffer* 02 estão de acordo com os valores esperados. A sinalização dos testes é idêntica à descrita na seção anterior.

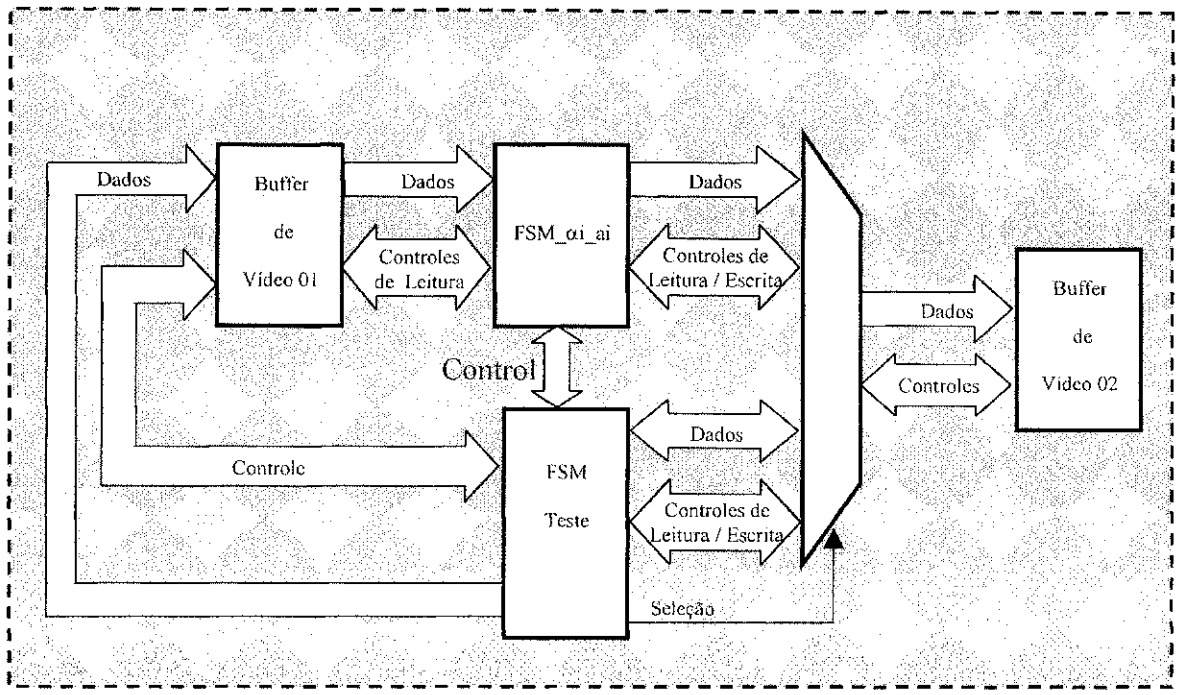


Figura 4-23: Diagrama em Blocos para Validação do Gerador de Parâmetro $\hat{\alpha}_i\_a_i$.

4.6.5 Validação do Gerador de Parâmetro $C\hat{D}_i\_b_i$

O diagrama em blocos da plataforma de testes utilizada para validação do Gerador de Parâmetros $C\hat{D}_i\_b_i$ é o mesmo utilizado para o Gerador de Parâmetros $\hat{\alpha}_i\_a_i$. O *buffer* 01 contém os valores dos *pixels* do decodificador de vídeo e o *Buffer* 02 os valores médios dos *pixels* e os valores acumulados do parâmetro b_i. Quando o processamento é do parâmetro $C\hat{D}_i$, a máquina de estados de teste seleciona o Gerador de Parâmetros $C\hat{D}_i\_b_i$

para calcular o $C\hat{D}_i$ e a medida que os valores são calculados, a máquina de estados de teste armazena os resultados em um *buffer* interno. Ao término do processamento, estes resultados são comparados com os valores esperados. Quando o teste é do parâmetro b_i, a máquina de estados de teste habilita o Gerador de Parâmetros $C\hat{D}_i\_b_i$ para calcular os valores do parâmetro b_i e ao final do processamento, verifica se os valores do parâmetro b_i armazenados no *buffer* de vídeo 02 estão de acordo com os valores esperados. A sinalização dos testes é idêntica à descrita na seção anterior.

4.6.6 Validação do Gerador de Histograma e Limiar

Na Figura 4-24 é mostrado o diagrama em blocos da plataforma de testes utilizada para validar o Gerador de Histograma e Limiar. Basicamente, a máquina de estados de teste habilita o Gerador de Histograma e Limiar e gera os valores do parâmetro $C\hat{D}_i$ para o processamento. Ao término do processamento, a máquina de estados de teste habilita o Gerador de Histograma e Limiar para calcular o limiar de comparação e, em seguida, verifica se está de acordo com o valor esperado.

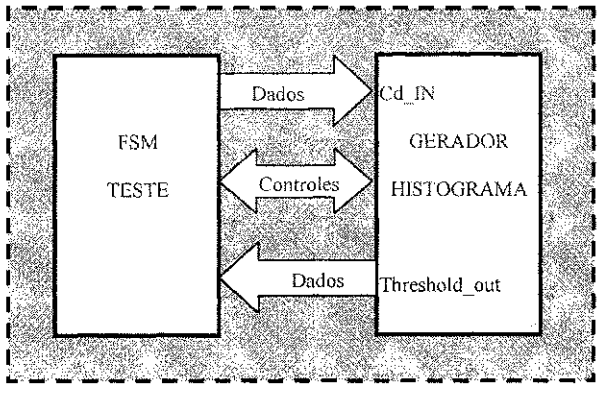


Figura 4-24: Diagrama em Blocos para Validação do Gerador de Histograma e Limiar.

4.6.7 Validação da Segmentação

Na Figura 4-25 mostra-se o diagrama em blocos da plataforma de teste utilizada para validar o módulo de segmentação. O módulo de segmentação classifica o *pixel* de acordo com os valores dos parâmetros $C\hat{D}_i$, $\hat{\alpha}_i$ e do limiar de comparação do histograma. A máquina de estados de testes gera os valores destes parâmetros na entrada do módulo de segmentação e verifica se a operação está correta.

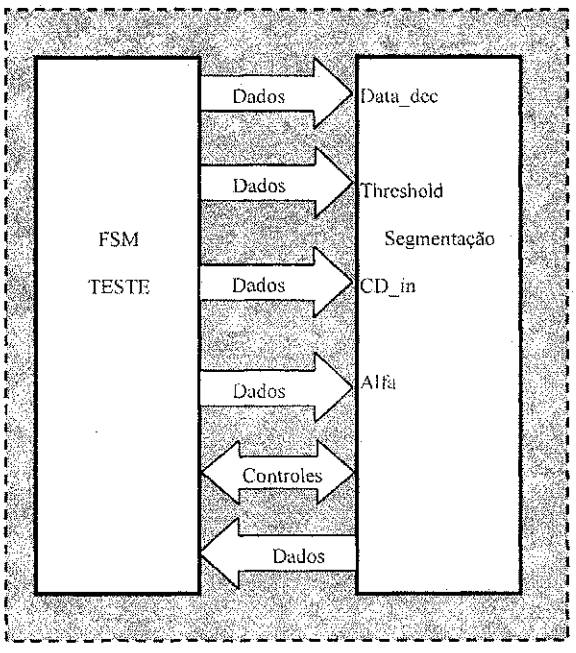


Figura 4-25: Diagrama em Blocos para Validação do Módulo Segmentação.

4.6.8 Validação da Interface de Acesso à Memória (IAM) e Controlador da IAM

O diagrama em blocos da plataforma utilizada para validação dos módulos IAM e Acesso a IAM é mostrado na Figura 4-26.

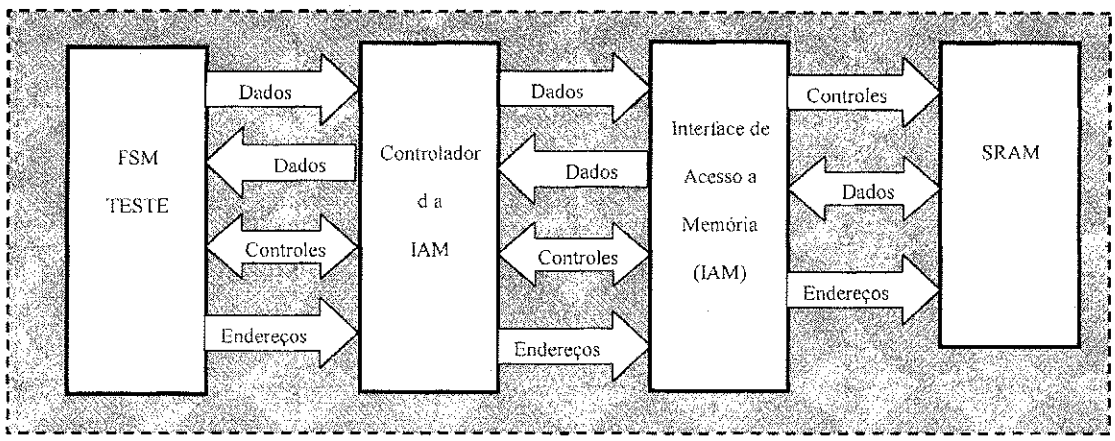


Figura 4-26: Diagrama em Blocos Para Validação dos módulos IAM e Controlador de Acesso à IAM.

O procedimento de teste consiste basicamente em escrever e ler dados na memória externa através de pedidos de interrupção para Controlador da Interface de Acesso à Memória. Em seguida, procede-se com a verificação dos valores armazenados. Outro teste realizado consiste na escrita e leitura de dados de modo alternado, iniciando na posição 00 ou 01 do banco de memória. A sinalização é feita de modo análogo ao descrito na seção anterior. A seleção do teste (seqüencial ou alternado) é feita por uma chave externa contida na plataforma.

Uma vez validados o Controlador da IAM e a Interface de Acesso à Memória, pode-se validar a interface com o decodificador de vídeo e a interface com o codificador de vídeo.

4.6.9 Validação da Interface Decodificador de Vídeo

A plataforma de teste utilizada para validação da Interface com o Decodificador de Vídeo é mostrada na Figura 4-27. Neste caso, os vetores de teste de entrada foram o obtidos de um sinal de vídeo de um gerador de padrões conectado à placa do decodificador de vídeo. A máquina de estados de teste controla a interface com o decodificador de vídeo que adquire os *pixels* do sinal de vídeo digitalizado para o formato RGB. Os *pixels* são armazenados na memória externa e o controle da IAM também é efetuado pela máquina de estados de teste. Após armazenar um quadro completo, a máquina de estados de teste efetua a leitura dos *pixels* armazenados para verificar se a Interface Decodificador de Vídeo efetuou corretamente a aquisição dos *pixels*, uma vez que a IAM e o Controlador da IAM já foram validados anteriormente.

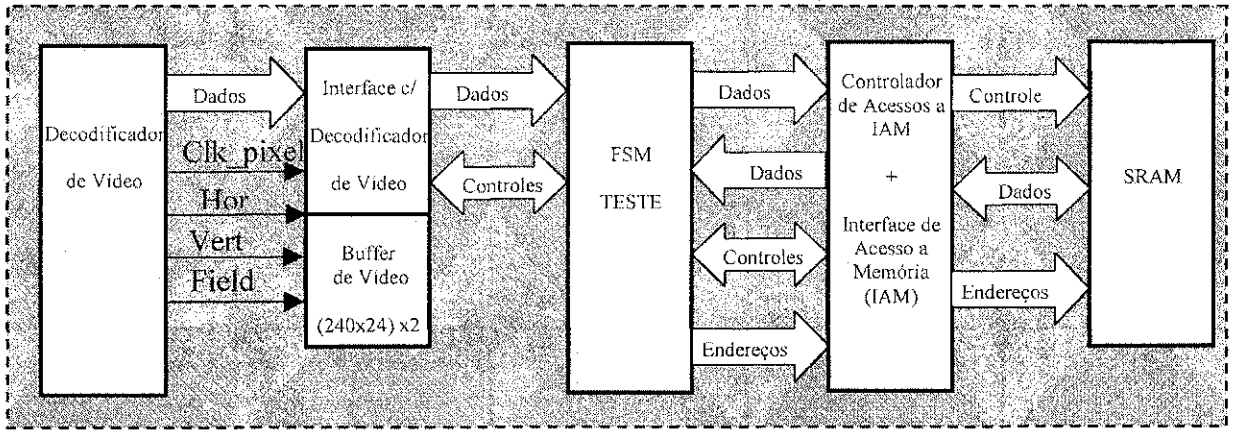


Figura 4-27: Diagrama em Blocos para Validação da Interface com o Decodificador de Vídeo.

4.6.10 Validação da Interface Codificador de Vídeo

Este foi o primeiro teste do sistema em tempo real. Consiste basicamente em capturar os *pixels* digitalizados em formato RGB e armazená-los na memória. Os *pixels* armazenados são lidos pela Interface Codificador de Vídeo e enviados para o codificador de vídeo. Este teste valida não somente a Interface Codificador de vídeo, mas, sobretudo o controle de acesso a IAM, pois neste caso têm-se dois blocos acessando a IAM, o de teste e a interface com o codificador de vídeo. Neste caso, a comparação foi efetuada com os sinais de vídeo de entrada (analógico), gerado pelo Gerador de padrões e o sinal de saída do Codificador de Vídeo.

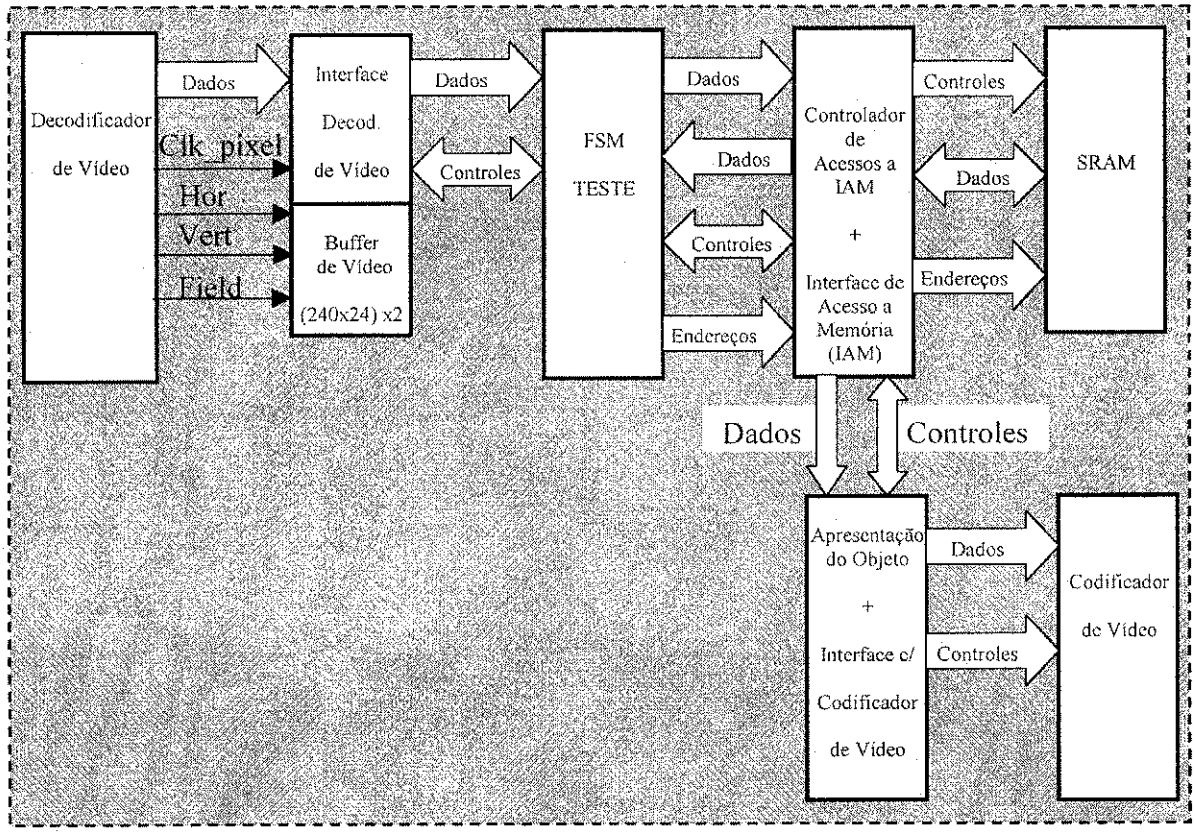


Figura 4-28: Diagrama em Blocos para Validação da Interface com Codificador de Vídeo.

4.6.11 Validação dos Módulos Integrados

Uma vez validados todos os módulos anteriores, tanto na simulação quanto fisicamente na plataforma (GPIP01), foi iniciada a integração dos módulos com a Fsm_Principal responsável por comandar cada fase das etapas de aprendizado e segmentação da imagem. Primeiramente, fez-se a validação comportamental do sistema contido na FPGA01, fixando as respostas (sinais de controle) esperadas do sistema contido na FPGA02. Com isso, validou-se a Fsm_Principal controlando os módulos da FPGA01. Em seguida, os módulos da FPGA01 foram interligados com os módulos da FPGA02 e procedeu-se com a depuração do sistema completo. Uma vez que o sistema estava idealmente funcionando procedemos com a simulação com atrasos visando identificar os problemas gerados pelo leiaute do sistema. Neste caso, constatou-se uma limitação de freqüência de operação dos dispositivos da plataforma GPIP01, restringindo o desempenho do sistema de segmentação. Quando o sistema foi simulado com atrasos operando em 66MHz, o atraso entre os sinais de controle internos era maior que ½ ciclo de *clock*, conforme ilustrado na Figura 4-29. Note que o sinal de controle denominado *line_start* da Interface Decodificador de Vídeo muda de estado somente após mais de ½ ciclo do *clock* do sistema (*clock_out).*

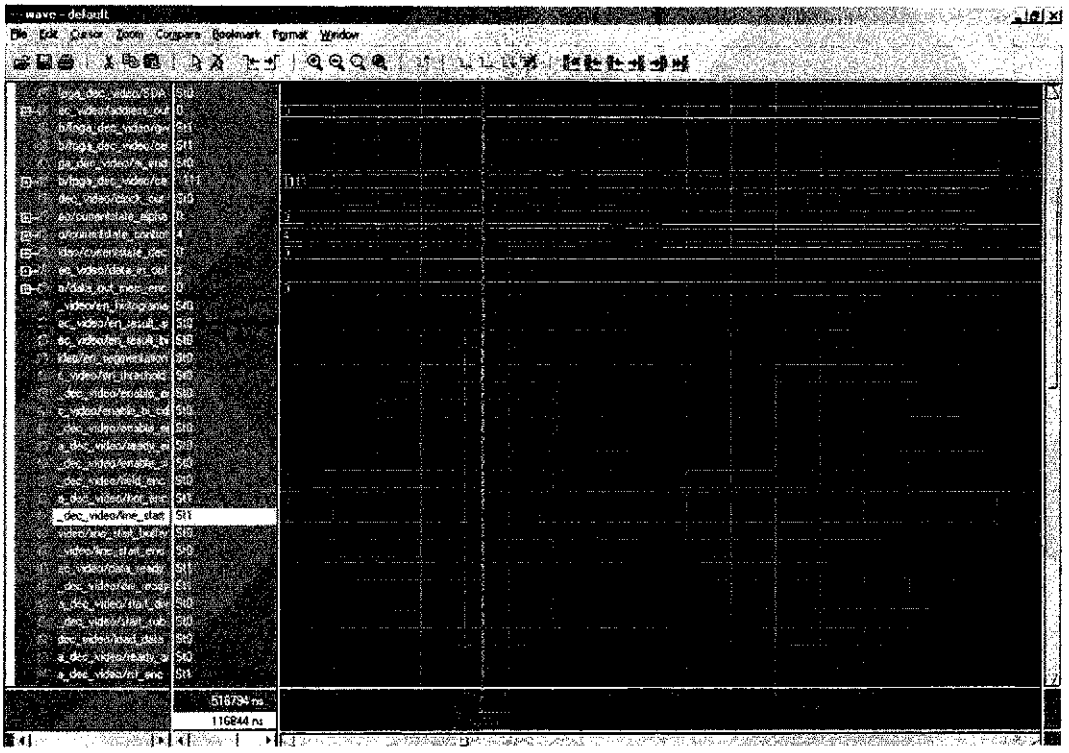


Figura 4-29: Atraso entre o sinais Line_start e Clock_out em 66MHz.

Constatou-se também que os atrasos eram diferentes entre os módulos do sistema. Por exemplo, o sinal que habilita o Gerador de Parâmetros $E_i\_S_i$, denominado, *enable_ei* apresentou um atraso menor que ½ ciclo de *clock*, conforme ilustrado na Figura 4-30.

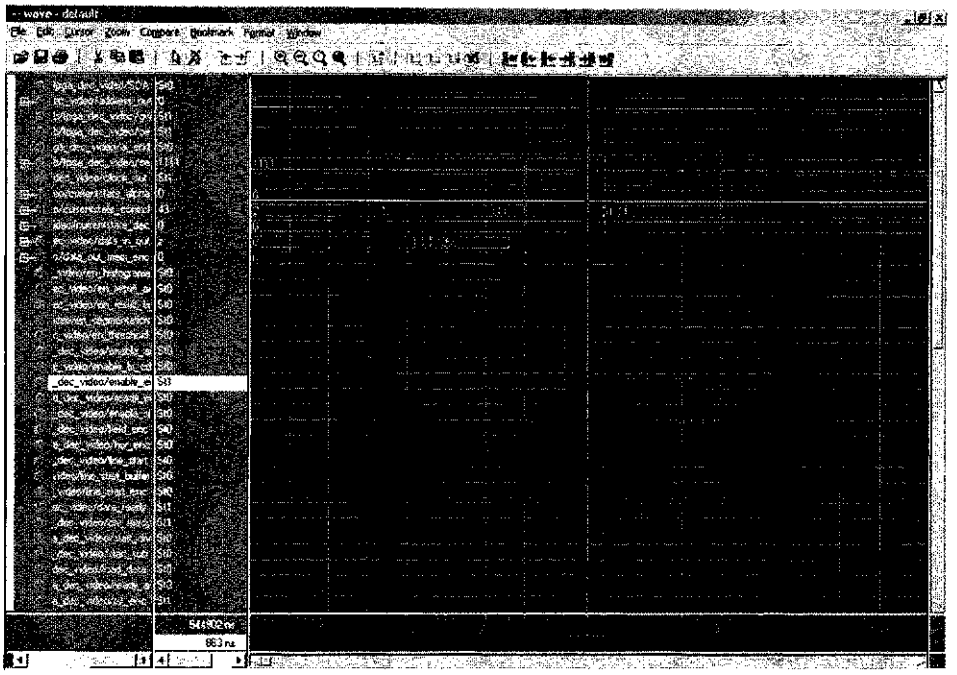


Figura 4-30: Atraso entre os sinais enable_ei e Clock_out em 66MHz.

Mediante estes resultados, a freqüência de *clock* do sistema foi reduzida para 54MHz, e, por conseguinte, a resolução horizontal para 180 *pixels* por linha. Operando nesta freqüência, ainda foram constatados problemas em função de atrasos, sobretudo, máquinas de estados mudando para estados inesperados. Em função destes problemas, a freqüência de operação do sistema foi reduzida para 40MHz e com isso a resolução horizontal diminuiu para 120 *pixels* por linha. Além da redução da freqüência de operação, alguns sinais de controle foram registrados em função de espúrios contidos nestes sinais. Em função da redução da resolução horizontal e, por conseguinte, do número de elementos de imagem analisados em cada linha, o número de quadros para processamento do parâmetro S_i e do histograma foi modificado para 54 e 64, respectivamente. Outra modificação realizada em função dos atrasos no sistema foi a transferência do módulo responsável pela transferência dos parâmetros da FPGA 01 para a FPGA02. Inicialmente este módulo estava na FPGA 01, porém apresentou problema na simulação com atrasos, sobretudo, mudanças de estados equivocados. Vale salientar que esta redução de resolução em função da freqüência de operação é uma limitação da plataforma GPIP01 que, em função da quantidade de elementos lógicos utilizados para implementação do sistema de

segmentação, não consegue operar em freqüência superiores a 40MHz que possibilitaria uma resolução superior a 120 *pixels* por linha. Esta limitação dos dispositivos lógicos programáveis está relacionada com a sua estrutura interna que além das células para implementação há também uma lógica adicional para programação das estruturas e isto limita a freqüência de operação do sistema. Em um ASIC não há estruturas para programação e o posicionamento dos módulos pode ser otimizado visando atingir a freqüência de operação desejada.

Uma vez solucionados os problemas de atrasos no sistema, iniciaram-se os testes do sistema de extração de objetos em um ambiente real. Os resultados obtidos com a implementação em *hardware*, em um ambiente real, são apresentados no próximo capítulo.

4.7 Conclusões

Neste capítulo foi apresentado o fluxo de projetos e o sistema desenvolvido para realizar a extração de objetos segundo o método estatístico de Horprasert. Resumidamente, as etapas seguidas para o desenvolvimento do sistema foram as seguintes: Definição do Sistema, Modelagem em HDL, Validação Estrutural e Comportamental, Síntese Lógica e Leiaute.

Apresentaram-se os módulos implementados no sistema, o princípio de funcionamento e os principais problemas e soluções durante a depuração do sistema totalmente integrado. A primeira limitação identificada na Plataforma de desenvolvimento foi durante a definição do sistema em que foi constatada a necessidade de dois dispositivos lógicos programáveis para elaboração do sistema de extração de objetos. A segunda limitação diz respeito à freqüência de operação que foi limitada a 40MHz em função dos atrasos internos nos dispositivos de lógica programável. A conseqüência da limitação da freqüência de operação foi a redução da resolução horizontal de 240 para 120 *pixels* por linha. Apesar destas limitações ainda foi possível avaliar o desempenho do método estatístico de Horprasert implementado em *hardware*.

5 Análise dos Resultados

Neste capítulo são apresentados os resultados obtidos do sistema de extração de objetos implementado em *hardware* em um ambiente real que apresenta alguns fenômenos tais como sombras, similaridade entre as cores dos objetos da imagem principal e da imagem de fundo e variação de iluminação. Além disso, o número de elementos lógicos utilizado no sistema bem como algumas formas de ondas do sistema físico também são apresentadas.

5.1 Avaliação da Extração de Objetos

Para avaliação do desempenho do sistema de extração de objetos implementado em *hardware*, são apresentadas 06 situações em um ambiente real com sombras e imagem de fundo heterogênea. As seqüências de imagens apresentadas são: a) imagem de fundo original, b) imagem de fundo original com o objeto a ser extraído, c) imagem resultante da extração sem a presença do objeto e d) imagem resultante da extração com a presença do objeto.

5.1.1 Avaliação da Imagem I

Na Figura 5-1 são mostradas as seqüências de imagens utilizadas para avaliação do sistema quanto a sombras geradas na imagem de fundo. A imagem de fundo original, contém sombras tanto na parede quanto no chão que são geradas por duas mesas localizadas nas extremidades da imagem de fundo. Há também pontos com um branco saturado em função da iluminação do ambiente. O objeto a ser extraído também gera uma sombra na imagem de fundo. Na imagem resultante, ainda sem a presença do objeto a ser

extraído, identificam-se pontos que são classificados como pertencentes a imagem do primeiro plano em função de erro estabelecido durante a determinação do limiar e, sobretudo em função de ruídos presentes no sinal digitalizado pelo decodificador de vídeo. Em uma seção posterior, é analisado detalhadamente o ruído gerado pelo decodificador de vídeo. Na Figura 5-1d mostra-se o resultado da extração do objeto. Apesar da baixa resolução do sistema, imposta pela limitação da Plataforma GPIP01, e das falhas nos pontos escuros do objeto, já previsto no modelo de Horprasert, a cadeira foi extraída da imagem de fundo apesar das sombras geradas pela cadeira e pelas mesas próximas da parede da imagem de fundo.

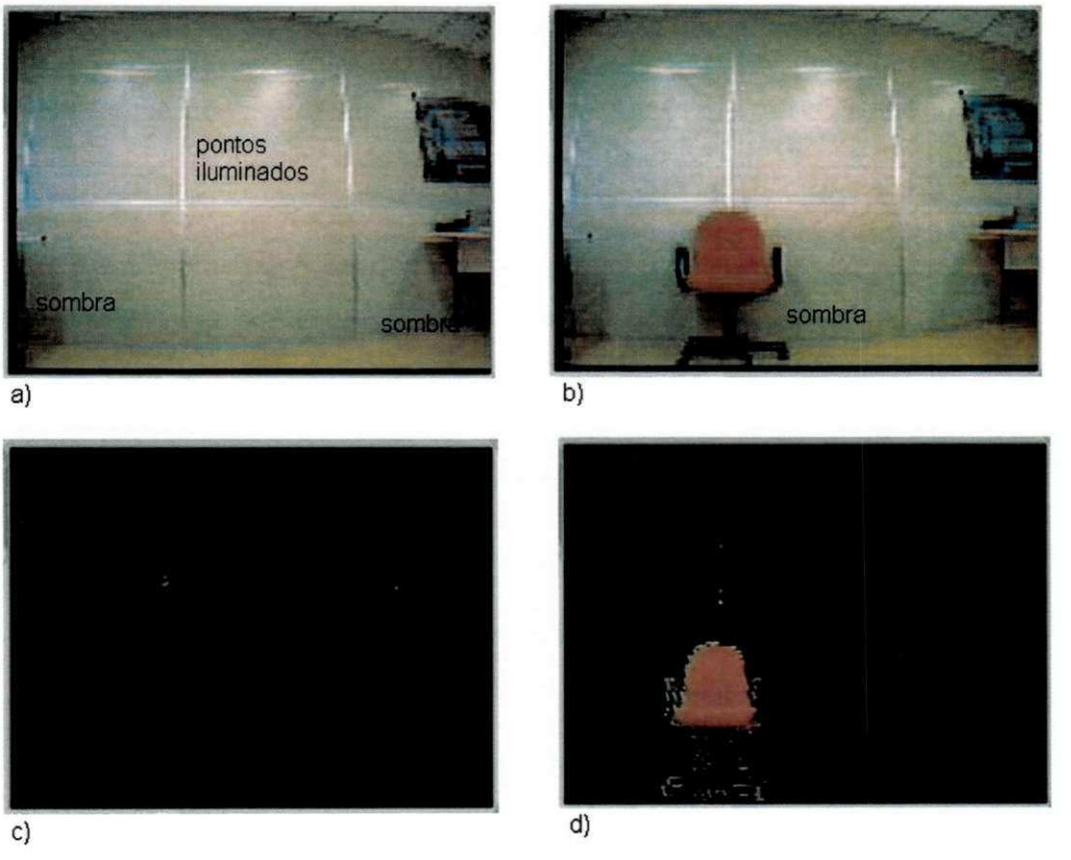


Figura 5-1: Seqüência de imagens para avaliação do Sistema de Extração de Objetos quanto a robustez a sombras na imagem de fundo: a) imagem de fundo original; b) imagem de fundo original com o objeto a ser extraído; c) imagem resultante da extração sem a presença de objeto; d) imagem resultante da extração com a presença do objeto.

5.1.2 Avaliação da Imagem II

Na Figura 5-2 são mostradas as seqüências de imagens utilizadas para avaliação do sistema quanto a similaridade de cor entre a imagem de fundo e o objeto da imagem principal. A imagem de fundo é a mesma utilizada na seção 5.1.1. Neste caso os objetos estão afastados da parede e, portanto, não geram sombras na imagem de fundo. Os resultados obtidos sem a presença dos objetos é similar ao obtido na Figura 5-1, isto é, alguns pontos são equivocadamente classificados como objeto da imagem do primeiro plano. Quanto à extração apesar da similaridade da cadeira do lado direito com o fundo, o sistema ainda foi capaz de efetuar a extração com poucos erros.

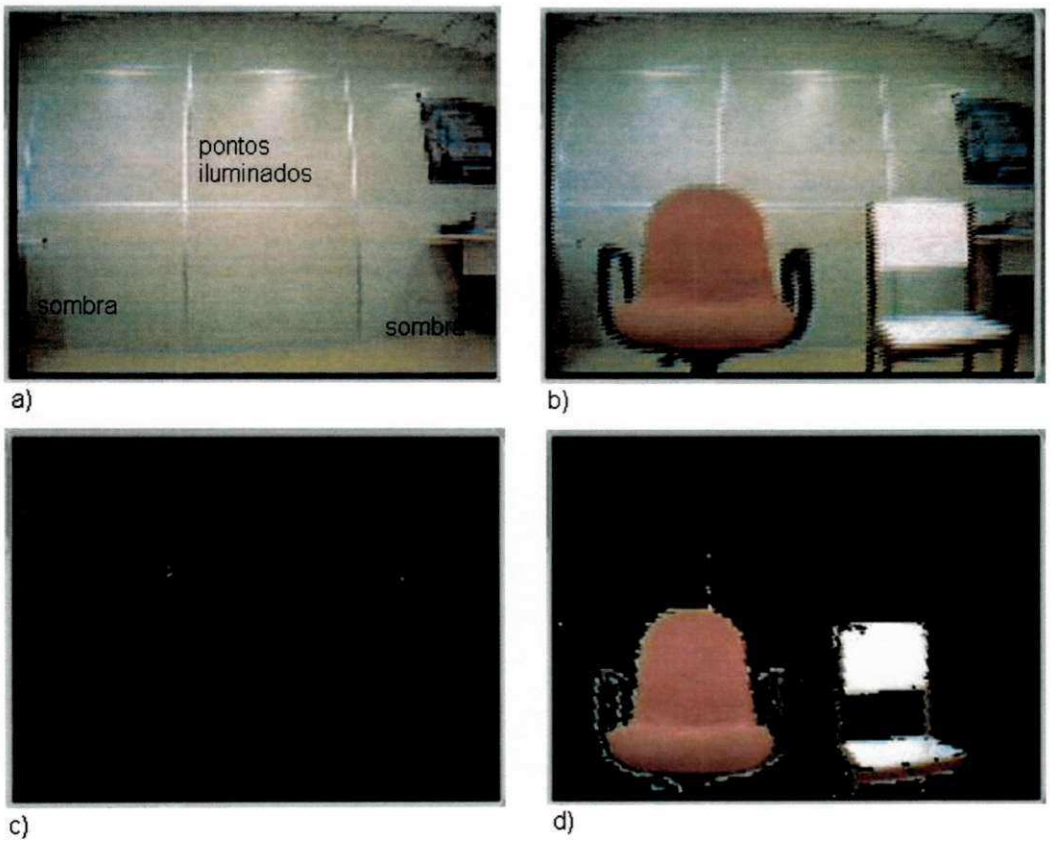


Figura 5-2: Seqüência de imagens para avaliação do Sistema de Extração de Objetos quanto a Similaridade de cor entre a imagem de fundo e o objeto do primeiro plano: a) imagem de fundo original; b) imagem de fundo original com o objeto a ser extraído; c) imagem resultante da extração sem a presença de objeto; d) imagem resultante da extração com a presença do objeto.

5.1.3 Avaliação da Imagem III

Na seqüência de imagens apresentadas na Figura 5-3, é avaliada a capacidade do sistema em extrair um objeto de um fundo heterogêneo. Para isto, inserimos duas cadeiras com cores diferentes, na imagem de fundo. Além disso, o objeto utilizado para extração tem a cor praticamente igual à de um dos objetos da imagem de fundo. Na imagem resultante da segmentação, observa-se que uma parte do objeto que estava em frente do objeto pertencente à imagem de fundo foi classificada como imagem de fundo em função da similaridade entre suas cores.

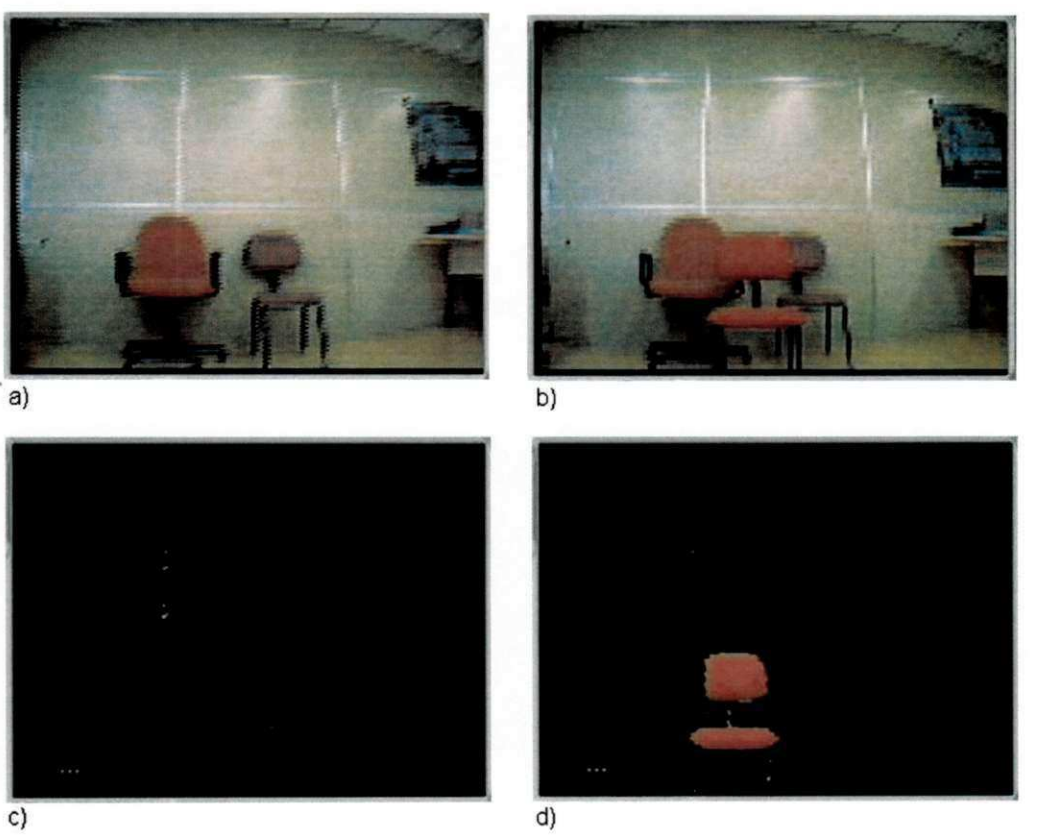


Figura 5-3: Seqüência de imagens para avaliação do Sistema de Extração de Objetos quanto a segmentação de objetos de uma imagem de fundo heterogênea: a) imagem de fundo original; b) imagem de fundo original com o objeto a ser extraído; c) imagem resultante da extração sem a presença de objeto; d) imagem resultante da extração com a presença do objeto.

5.1.4 Avaliação da Imagem IV

O objetivo da seqüência de imagens apresentadas na Figura 5-4 é avaliar o sistema em uma situação em que uma pessoa for utilizada no processo de segmentação. Pode-se observar que há falhas nos contornos dos braços e na parte inferior em função de ruídos existentes no sinal digitalizado e, além disso, em função da baixa resolução temos poucos elementos de imagem em cada linha e a perda de alguns deles torna-se bastante significativa na imagem resultante da extração.

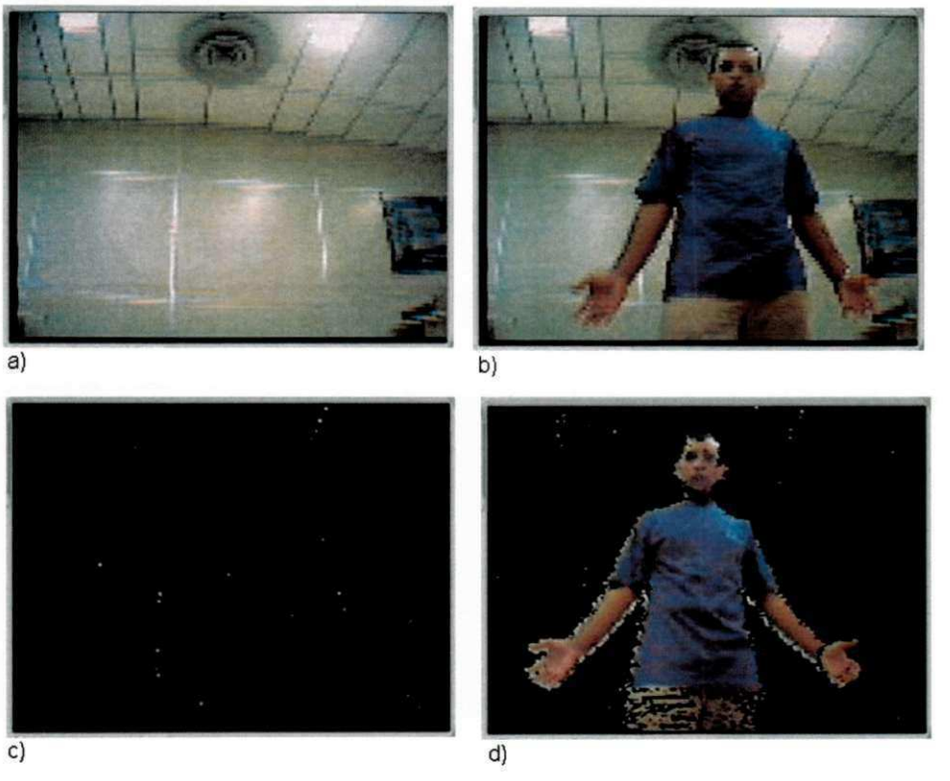


Figura 5-4: Seqüência de imagens para avaliação do Sistema de Extração de Objetos quanto a detecção de uma pessoa na imagem de primeiro plano: a) imagem de fundo original; b) imagem de fundo original com o objeto a ser extraído; c) imagem resultante da extração sem a presença de objeto; d) imagem resultante da extração com a presença do objeto.

5.1.5 Avaliação da Imagem V

Na seqüência de imagens da Figura 5-5 a cor da camisa da pessoa inserida para extração é bem distinta da cor da imagem de fundo e observa-se que a imagem final extraída está com falhas apenas nos contornos em função, sobretudo, das transições abruptas de cor geradas nestas regiões. O sensor da câmera, nas regiões de contorno, pode detectar a imagem de fundo ou o objeto e isto gera um ruído no sinal capturado.

Figura 5-5: Seqüência de imagens utilizada para avaliação do Sistema de Extração de Objetos quanto a segmentação de cores distintas da imagem de fundo: a) imagem de fundo original; b) imagem de fundo original com o objeto a ser extraído; c) imagem resultante da extração sem a presença de objeto; d) imagem resultante da extração com a presença do objeto.

5.1.6 Avaliação da Imagem VI

Nas Figura 5-6a e Figura 5-6b são mostradas as imagens resultantes da extração, sem a presença de objetos, após a variação da iluminação do ambiente. Na Figura 5-6a, a iluminação da imagem de fundo diminuiu abruptamente em função do desligamento de quatro lâmpadas fluorescente de 40W que estavam iluminando a imagem de fundo na etapa de aprendizado. A sala onde os testes foram realizados contém 28 lâmpadas fluorescentes de 40W cada. Na Figura 5-6b, a imagem de fundo foi iluminada após a etapa de aprendizado. Neste caso, duas lâmpadas fluorescentes que estavam desligadas no momento do aprendizado da imagem de fundo foram ligadas. Na Figura 5-6c mostra-se o resultado da extração do objeto após a diminuição da iluminação da imagem de fundo e na Figura 5-6d mostra-se o resultado da extração do objeto com o aumento da iluminação da imagem de fundo.

Em função da variação abrupta da iluminação alguns elementos de imagem pertencentes à imagem de fundo são classificados constantemente com pertencentes à imagem do primeiro plano. No entanto, a segmentação do objeto não fora prejudicada. Na realidade estas variações a que o sistema foi submetido são muito intensas e o modelo de segmentação está preparado para pequenas variações globais e locais na imagem. Apesar disso, observa-se que mesmo nesta situação extrema o sistema ainda foi capaz de proceder com a extração do objeto da imagem do primeiro plano, ou seja, pequenas variações na iluminação são praticamente toleráveis pelo sistema.

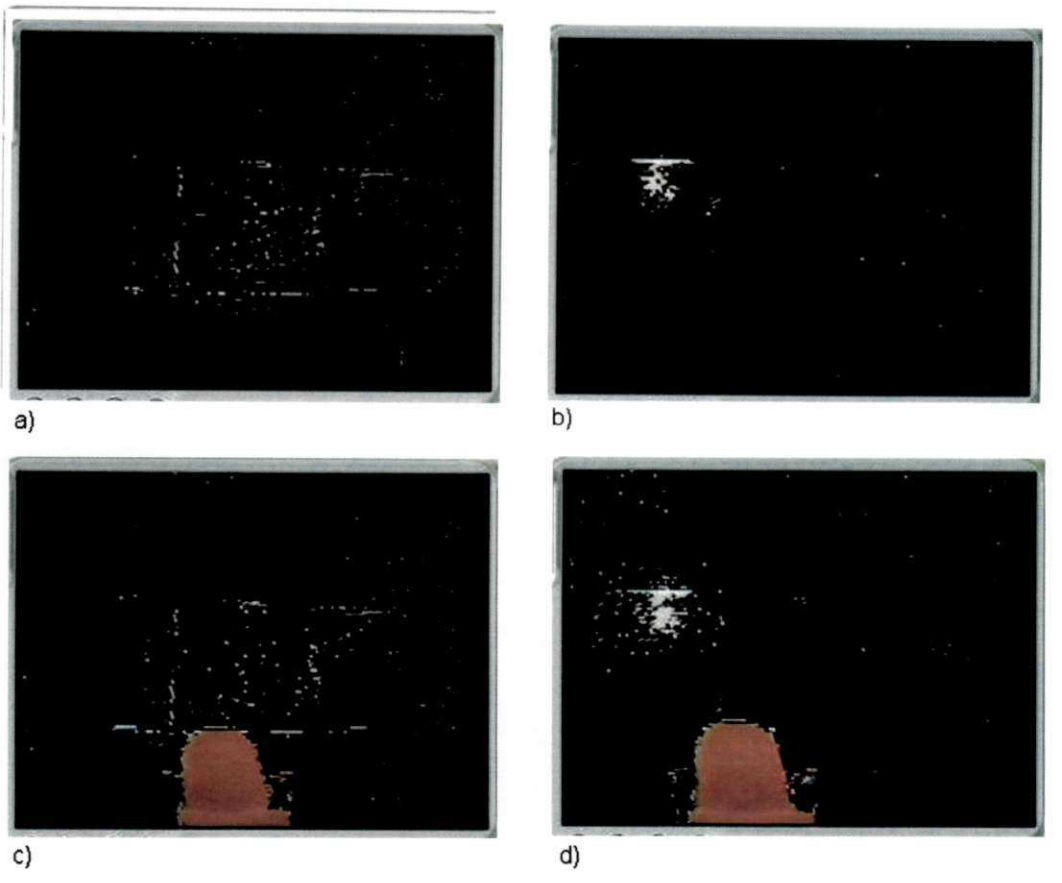


Figura 5-6: Seqüência de Imagens para avaliação do sistema de extração quanto a variações bruscas de iluminação: a) imagem resultante da extração, com a diminuição da iluminação, sem a presença de objetos; b) imagem resultante da extração, com o aumento da iluminação, sem a presença de objetos; c) imagem resultante da extração do objeto com a diminuição da iluminação; d) imagem resultante da extração do objeto com o aumento da iluminação;

5.2 Sinal do Decodificador de Vídeo

Conforme mencionado anteriormente, constatou-se a presença de ruídos no sinal digitalizado pelo decodificador de vídeo. Isto afeta o desempenho do sistema de extração, tanto na etapa de aprendizado quanto na fase de extração do objeto, visto que um elemento de imagem com ruído ora é classificado como imagem de fundo e em outro instante, em função da mudança de valor, pode vir a ser classificado como pertencente ao objeto da imagem do primeiro plano. O resultado, são pontos na imagem resultante variando constantemente. Para constatarmos a presença e o efeito destes ruídos no processamento

foram feitas algumas medições elétricas do sinal digitalizado pelo decodificador de vídeo. A seguir é descrito este método.

5.2.1 Medição Elétrica

Para se analisar o sinal digitalizado pelo decodificador de vídeo, foi aplicado um sinal de vídeo na sua entrada com padrão branco (1Vpp) e o bit mais significativo e o bit menos significativos, de uma das componentes de saída do decodificador de vídeo, foram medidos. As formas de ondas dos sinais estão ilustradas na Figura 5-7.

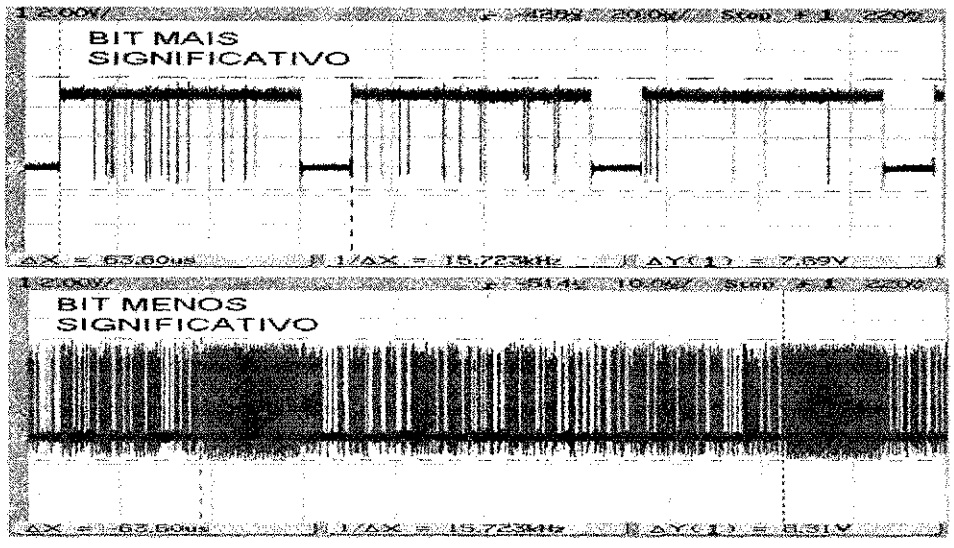


Figura 5-7: Bits mais significativo e menos significativo da componente R na saída do decodificador de Vídeo para um sinal de vídeo de entrada igual a 1 Vpp.

O sinal esperado na saída do decodificador de vídeo seria todos os bits sempre em nível lógico 1, uma vez que uma imagem branca resulta no valor máximo do conversor que neste caso é de 8 bits e, portanto, a saída deveria ser 255. Entretanto, o que observamos são sinais espúrios presentes na saída, sendo que no bit menos significativo os espúrios são bem mais intensos. Em função destes espúrios, o sinal digitalizado está equivocadamente assumindo valores correspondentes a cores completamente diferentes da entrada. O método utilizado para extração de objetos implementado neste trabalho utiliza a cor para distinção entre um objeto da imagem de fundo e da imagem do primeiro plano e com esta variação a

classificação é extremamente prejudicada. A causa destes ruídos é intrínseca à placa do decodificador de vídeo. A solução ideal seria desenvolver um outro leiaute da placa do decodificador de vídeo, no entanto isto está fora do escopo deste trabalho. Para minimizar o nível de ruído digitalizado, o nível do sinal de vídeo na entrada foi aumentado de 1 Vpp para 1,5 Vpp. Na Figuras 5-8 mostram-se os resultados obtidos com esta modificação. O bit mais significativo ficou praticamente sem ruídos e o bit menos significativo ainda está com ruído, porém o sinal está caracterizado. Convém salientar que os resultados apresentados na seção 5-1 foram obtidos com um nível de sinal de entrada igual a 1 Vpp. O nível do sinal de vídeo de 1,5 Vpp é o valor máximo especificado para a entrada do decodificador de vídeo e desta maneira evitou-se testes exaustivos com este nível de sinal. Além disso, é recomendado utilizar-se 1 Vpp com 75 ohms de impedância de entrada para o decodificador de vídeo. Portanto, a solução mais adequada para eliminação dos ruídos no sinal digitalizado seria a elaboração de um outro leiaute da placa do decodificador de vídeo.

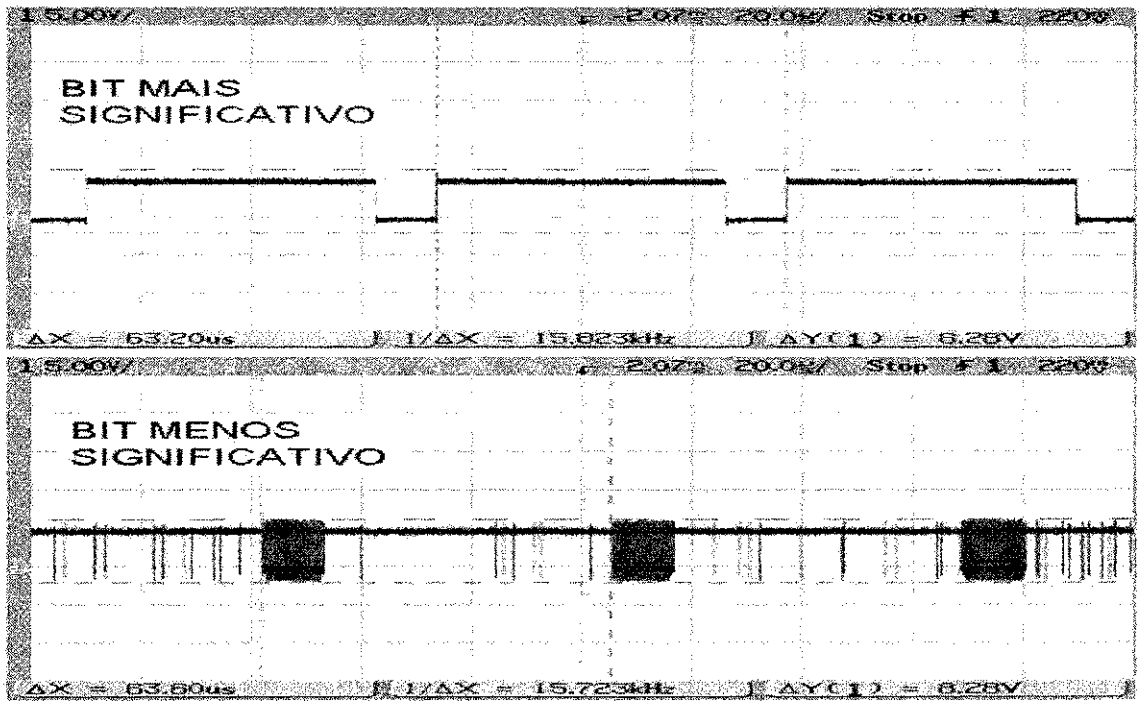


Figura 5-8: Bits mais significativo e menos significativo da componente R na saída do decodificador de Vídeo para um sinal de vídeo de entrada igual a 1,5 Vpp.

5.3 Determinação do $\tau_{\alpha lo}$

Conforme já mencionado, o modelo de extração de objetos apresenta uma limitação na detecção de objetos escuros. Para minimizar este problema, Horprasert [4] estabeleceu um limiar de comparação ($\tau_{\alpha lo}$) baseado no valor do parâmetro $\hat{\alpha}_i$ e, desta maneira, objetos com distorção de cor maior que o limiar estabelecido no histograma ou com distorção de brilho menor que $\tau_{\alpha lo}$ são classificados como objetos da imagem do primeiro plano. As avaliações realizadas anteriormente foram feitas apenas comparando-se a distorção de cor de *pixel* da imagem com o limiar do histograma. Na Figura 5-8 é apresentada uma seqüência de imagens utilizadas para avaliar o sistema de extração quanto a detecção de um objeto escuro utilizando-se apenas a comparação da distorção de cor de cada *pixel*.

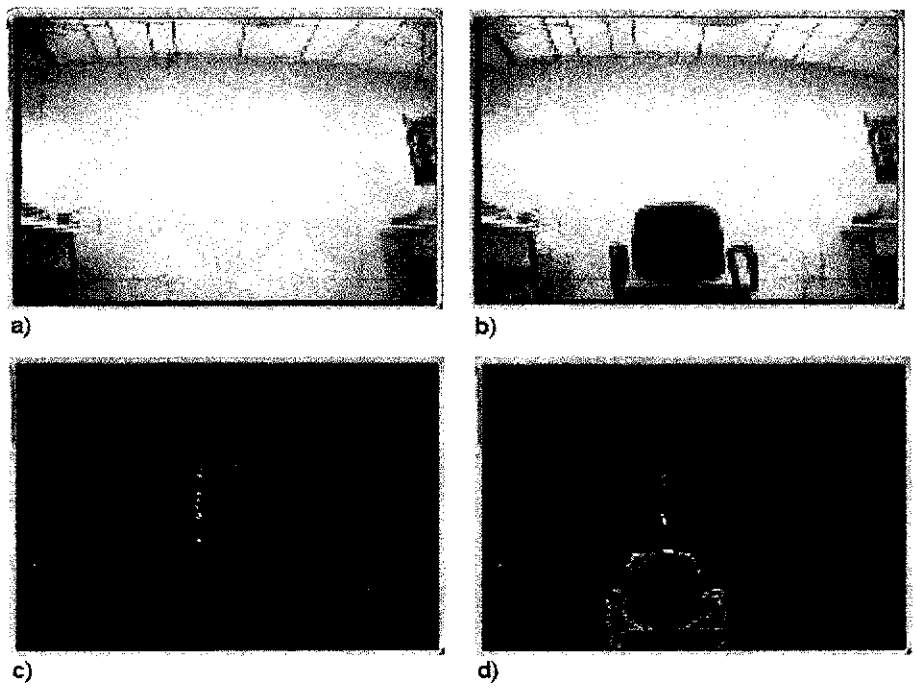


Figura 5-9: Seqüência de Imagens utilizada para avaliar o sistema de extração quanto a detecção de objetos escuros utilizando apenas a distorção de cor: a) imagem de fundo original; b) imagem de fundo original com o objeto a ser extraído; c) imagem resultante da extração sem a presença de objeto; d) imagem resultante da extração com a presença do objeto e sem o limiar de comparação da distorção de brilho.

De acordo com os resultados, observou-se que ocorre muito erro na detecção do objeto. No sistema implementado neste trabalho o τ_{alo} foi determinado empiricamente e os melhores resultados foram obtidos com τ_{alo} igual a 90. Na Figura 5-10 são apresentados os resultados obtidos com a utilização da distorção de cor e da distorção de brilho na segmentação. Os resultados obtidos são bem melhores que os anteriores e, com isso, os erros de segmentação para *pixels* com cor escura foram minimizados.

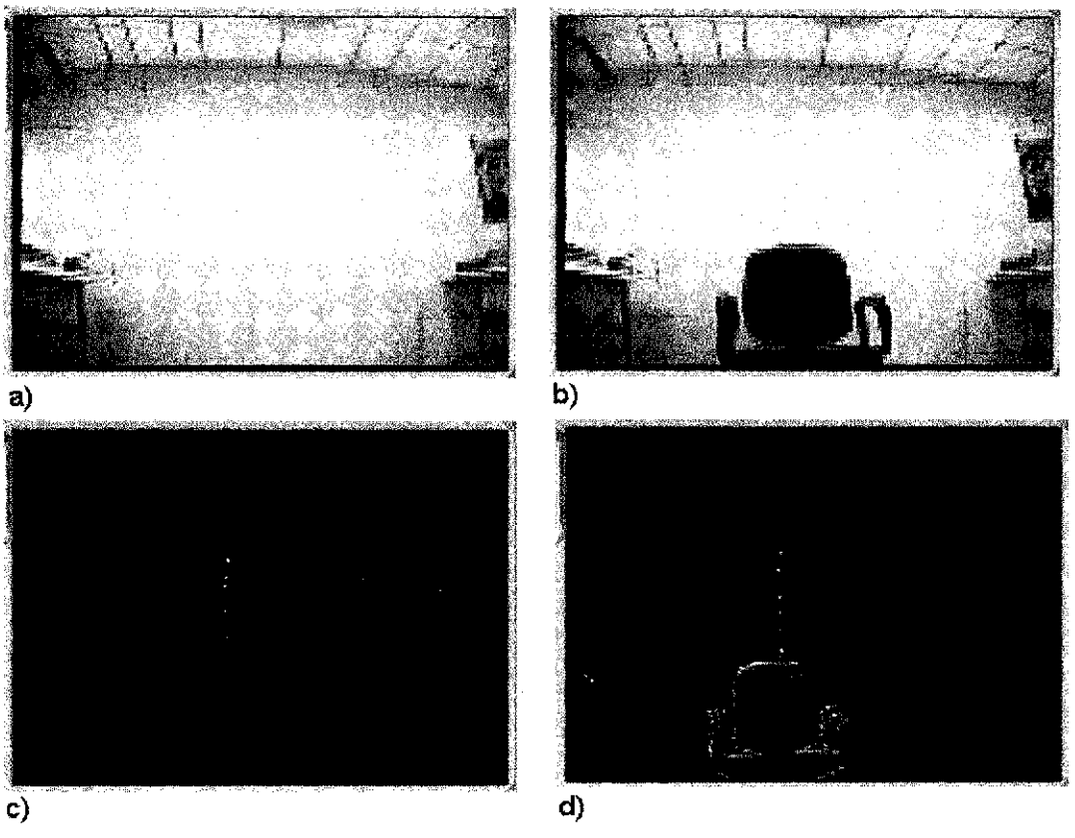


Figura 5-10: Seqüência de Imagens utilizada para avaliar o sistema de extração quanto a detecção de objetos escuros utilizando a distorção de cor e a distorção de brilho: a) imagem de fundo original; b) imagem de fundo original com o objeto a ser extraído; c) imagem resultante da extração sem a presença de objeto; d) imagem resultante da extração com a presença do objeto e com o limiar de comparação da distorção de brilho.

5.4 Resultados da Síntese Física

O sistema implementado neste trabalho está distribuído em dois dispositivos de lógica programável com 8320 elementos lógicos cada um. No dispositivo 01 foram implementados os seguintes módulos: Interface Decodificador de Vídeo, Gerador de Parâmetros $\hat{\alpha}_i\_a_i$, Gerador de Parâmetros $E_i\_S_i$, Fsm_Principal, *Buffer* de Parâmetros, Interface com Memória Externa, Multiplexador de Sinais de Controle, Interface de Acesso à Memória e a Interface I2C. Estes módulos utilizam 91% dos elementos lógicos disponíveis. No dispositivo 02 foram implementados os seguintes módulos: Interface Codificador de Vídeo, Segmentação, Gerador de Histograma e Limiar, Gerador de Parâmetros $C\hat{D}_i\_b_i$, *Buffer* de Parâmetros, Interface de Acesso à Memória, Controlador da Interface de Acesso à Memória, Interface com Memória, Transferência de Parâmetros e o Registrador de Parâmetros. Estes módulos utilizam 88% dos elementos lógicos disponíveis. Quanto maior for o número de elementos lógicos utilizados, o leiaute, torna-se mais difícil e, por conseguinte, os atrasos internos são maiores. Em função disso, a freqüência de operação obtida para o sistema foi de 40MHz.

5.5 Análise geral dos Resultados

Os objetivos estabelecidos para o escopo deste trabalho foram os seguintes: implementação de um método para extração de objetos de um fundo heterogêneo utilizando a plataforma GPIP01 visando atender aos requisitos do mercado de eletrônica de consumo, efetuar otimizações nos diagramas em blocos e máquinas de estados inicialmente propostas por Printes [7] e avaliar o desempenho do método de extração implementado em hardware.

Primeiramente, foi feita uma análise da quantidade de elementos lógicos necessários para a implementação do sistema de extração de objetos. Constatamos que o dispositivo lógico da plataforma GPIP01 não comportaria todo o sistema e uma nova arquitetura do

sistema foi proposta e implementada. Basicamente, o sistema foi distribuído em dois dispositivos lógicos programáveis. As otimizações realizadas foram, sobretudo, a unificação das máquinas de estado de controle dos blocos de cálculo relacionados, ou seja, os módulos de cálculo dos parâmetros E_i e S_i que são controlados pela mesma máquina de estados e compartilham alguns blocos, os módulos de cálculo dos parâmetros α_i e a_i que também são controlados por uma mesma máquina de estados assim como os módulos de cálculo dos parâmetros CD_i e b_i. Uma vez definida a nova arquitetura, procedeu-se com o desenvolvimento e depuração do projeto seguindo o fluxo de projetos citado no Capítulo 4.

A plataforma apresentou algumas limitações quanto à freqüência máxima de operação e também apresentou ruídos no sinal digitalizado que foram contornados sem comprometer o objetivo final deste trabalho. Em seguida, foi realizada a análise do desempenho do sistema de extração de objetos realizando testes em um ambiente real em situações que evidenciaram os principais fenômenos encontrados neste ambiente, tais como sombras, similaridade entre cores de objetos e variação de iluminação. O sistema funcionou adequadamente em tempo real, isto é, com imagens na taxa de 30 quadros por segundo e com isso mantive-se a sensação de continuidade da imagem tal como nos padrões atuais de televisão. Desta maneira, os objetivos estabelecidos nestes trabalhos foram atingidos e algumas melhorias e trabalhos futuros são propostos no Capítulo 6.

6 Conclusão

6.1 Resultados

Nesta dissertação continuou-se o trabalho realizado por Printes [7] que validou em MatLab um método para extração de objetos e apresentou uma plataforma para o desenvolvimento de sistemas de imagens. A fase implementada neste trabalho consistiu na implementação em hardware do método de extração validado no MatLab. A plataforma GPIP01 apresentou algumas limitações que implicaram na redução da resolução horizontal de 240 para 120 pixels por linha, porém isto não inviabilizou a análise do método de extração em hardware.

O sistema implementado mostrou-se eficiente quando submetido aos fenômenos normalmente encontrados nos ambientes reais, quais sejam sombras e variação de luminosidade. Constatou-se a limitação do método de extração nos casos em que o objeto é escuro, conforme já esperado, e identificou-se também ruídos no sinal de saída do decodificador de vídeo. O sistema executou o processamento todo em tempo real, isto é, processando imagens na taxa de 30 quadros por segundo mantendo as características dos padrões atuais de televisão.

Tendo implementado e validado o método de extração em hardware, foi possível provar e mostrar o conceito do sistema ser capaz de extrair um objeto de uma imagem de fundo heterogênea e estática, em tempo real. Com isso em trabalhos futuros pode-se desenvolver um sistema para inserção do objeto extraído em uma segunda imagem oriunda de uma outra fonte de vídeo, complementando a elaboração do sistema para extração de objeto e inserção em uma segunda imagem.

6.2 Trabalhos Futuros

O escopo do sistema implementado neste trabalho é a extração de objetos de uma imagem de fundo heterogênea, porém estática, em tempo real. O sistema implementado na plataforma GPIP01 apresentou algumas restrições que devem ser melhoradas para implementação do sistema final capaz de extrair um objeto de uma imagem de fundo heterogênea e inserí-lo em uma imagem oriunda de uma outra fonte de vídeo. As sugestões para trabalhos futuros serão divididas em melhorias do método de extração de objetos e implementações complementares.

6.2.1 Melhorias do Método de Extração de Objetos

No método de extração de objetos implementado neste trabalho, os limiares de comparação utilizados na segmentação da imagem, delimitam um cilindro em torno da linha de cromaticidade esperada. Os limiares de comparação são fixos ao longo da linha de cromaticidade e, desta maneira, objetos escuros sempre estarão situados dentro dos limites do cilindro e com isso o sistema classifica-os equivocadamente como sombras. Horprasert [4] estabeleceu um limiar de comparação mínimo baseado no nível da distorção de brilho para

contornar esta limitação. Entretanto, estabelecendo-se o limiar τ_{alo}, objetos escuros pertencentes a imagem de fundo também serão classificados com pertencentes à imagem de primeiro plano. A solução mais adequada para este problema do modelo computacional de Horprasert [4] seria estabelecer os limiares de comparação levando em consideração a distância do ponto em análise em relação à origem do sistema de coordenadas R, G e B. Com isso, os limiares de comparação delimitariam um cone ao longo da linha de cromaticidade esperada.

O método de extração de objetos implementado neste trabalho utiliza um modelo computacional baseado no plano RGB. Neste plano as componentes de cores R, G e B são

sensíveis a mudanças de iluminação. Pode-se desenvolver um novo método baseado no plano YIQ. Neste caso, o sinal de luminância (Y) contém apenas a variação de brilho da imagem e os sinais I e Q as informações de cores da imagem. Desta maneira, pode-se manter as características do método de Horprasert [4], isto é, a análise estatística dos elementos de imagens em um plano bidimensional de cores e a informação de brilho contida no sinal de luminância. O intuito deste novo método seria, sobretudo, obter um método robusto e com equações mais simples que as apresentadas no método de Horprasert [4]. Desta maneira, pode-se reduzir o número de elementos lógicos necessários para implementação do sistema e, por conseguinte, o seu custo final.

6.2.2 Implementações Complementares

Os trabalhos complementares possivelmente a serem realizados futuramente são os seguintes:

- Melhoramento da Plataforma GPIP01 com hardware para controle de uma câmera USB;
- Realização da etapa de inserção do objeto em uma segunda imagem;
- Implementação do ASIC;

7 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Gonzalez, Rafel C., Woods, Richard C., "Digital *Image Processing*", Addison-Wesley 1992.

[2] Bovik, A.L. "*Handbook of Image & Video Processing*", San Diego, Academic Press, 2000, pp.383

[3] L. Tonietto. "Análise de algoritmos para Chroma-key". Disponível em: http://www.inf.unisinos.br/~marcelow/ensino/tc/ckey/ckey.html. Acesso em: 19 de agosto de 2002.

[4] T. Horprasert, D. Harwood, L.S. Davis. "A *statistical approach for Real Time Robust background subtraction*". In IEEE ICCV'99 FRAME-RATE WORKSHOP. 1999.

[5] Seki, M., Fujiwara,H., Sumi,K. "A *Robust Background Subtraction Method for Changing Bachground*". Applications of Computer Vision, 2000, Fifth IEEE Workshop on., 2000, pp. 207 – 213.

[6] Y. Ivanov, A. Robick, J. Liu. "*Fast lighting independent background subtraction*". International Journal of Computer Vision, 37(2), pp.199-209, June 2000.

[7] Printes, André Luíz, "Circuito Integrado para Extração de fundo não homogêneo de imagens dinâmicas em tempo real" Dissertação de Mestrado, Universidade Federal de Campina Grande, Dezembro 2002.

[8] MatLab: Disponível em: http://www.mathworks.com/. Acesso em 04 de dezembro de 2002.

[9] Harville, M., Gordon, G., Woodfill, J. "*Foreground Segmentation Using Adaptive Mixture Models in Color and Depth*". Detection and Recognition of Events in Video, 2001, Proceedings IEEE Workshop on, 2001, pp. 3-11.

[10] K. Toyama, J.Krumm, B. Brumitt, B.Meyers. "*Wallflower: Principles and Practice of Background Maintenance*" in International Conference on Computer Vision, (Kerkyra, Greece), pp. 255-261, 1999.

[11] Y.H. Yang and M.D. Levine. "*The Background Primal Sketch: An Approach for Tracking Moving Objects*". Machine Vision and Applications, 5:17-34, 1932

[12] I. Dinstein. "*A New Technique for Visual Motion Alarm*". Pattern Recognition Letters, 8:346-351, 1989.

[13] T.J. Ellis, P. Rocín, and P. Golton. "*Model-based Vision for Automatic Alarm Interpretation*". IEEE Aerospace and Electronic Systems Magzine, 1991.

[14] Y.Z. Hsu, H.H. Nagel, and G. Rekers. "*New Likelihood Test Methods for Change Detection in Image Sequences*". Computer Vision, Graphics, and Image Processing, 1994.

[15] R. Jain. "*Extraction of Motion Information from Peripheral Processes*". IEEE Trans. Pattern Analysis and Machine Intelligence, 1981.

[16] D. Koller, J. Weber, and J. Malik. "*Robust Multiple Car Tracking with Occlusion-Reasoning*". In Europ. Conf. Computer Vision, 1994.

[17] A. Singh. "*Digital Change Detection Techniques Using Remotely-Sensed Data*". Int. J. Remote Sensing, 1989.

[18] T.F. Knoll, L.L. Brinkley, and E.J. Delp. "*Difference Picture Algorithms for the Analysis of Extracellular Components of Histological Images*". J. Histochem. Cytochem., 1985.

[19] Rosin, Paul L. "*Thresholding for Change Detection*". In sixth International Conference Computer Vision , 1998.

[20] Rosin, Paul L., Ellis, Tim. "*Image Difference Threshold Strategies and Shadow Detection*". Disponível em:
http://www.cs.cf.ac.uk/User/Paul.Rosin/resources/papers/shadows.pdf.
Acesso em 19 de janeiro de 2003.

[21] LI, Dalong. "*Moving Objects Detection By Block Comparison*". Electronics, Circuit and Systems, 2000. ICECS 2000. The 7th IEEE International Conference On, Volume 1, 2000.

[22] D. P Elias, N.G Kingsbury. "*An Efficient Block Segmentation Algorithm for True Motion Estimation*". Image Processing and Applications, 1997, Sixth International Conference On, Volume 1.

[23] H., Ismail, H.,David, D.S., Larry, "*W4: Real-Time Surveillance of People and Their Activities*". Disponível em:
http://www.umiacs.umd.edu/users/hismail/Publications/Proposal.pdf. Acesso em 19 de janeiro de 2003.

[24] KaewTrakullPong, P., Bowden, R. *"An Improved Adaptive Background Mixture Model Real-time Tracking with Shadow Detection"*. Disponível em: http://www.ee.surrey.ac.uk/Personal/R.Bowden/publications/avbs01/avbs01.pdf. Acesso em 19 de janeiro de 2003.

[25] S., Chris, W.E.L. Grimson, *"Adaptive Background Mixture Models for Real-time Tracking"*. Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.

[26] W.E.L. Grimson, C. Stauffer, R. Romano, L. Lee, *"Using adaptive tracking to classify and monitor activities in a site"*. Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on.

[27] Quartus. Disponível em: http://www.altera.com. Acesso em 21 de dezembro de 2002.

[28] Smith, Douglas J. " HDL CHIP DESIGN – *A Pratical Guide for Designing, Synthesizing and Simulations ASICs and FPGAs using VHDL or Verilog*. Capítulo 09, pag. 289.

[29] Tommiska, Matti T. " *Area-Efficient Implementation of a Fast Square Root Algorithm". Third IEEE International Conference on Devices, Circuits and Systems, March 2000,* Session 21C – *Digital Signal Processing*.

[30] Verilog. Disponível em: http://www.verilog.com. Acesso em 18 de dezembro de 2002.

[31] Leonardo Spectrum. Disponível em: http://www.mentor.com/leonardospectrum/. Acesso em 20 de dezembro de 2002.

[32] ModelSim. Disponível em: http://www.model.com. Acesso em 20 de dezembro de 2002.