UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Elétrica e de Computação

Departamento de Engenharia de Computação e Automação Industrial

FÁBIO LUÍS PICELLI LUCCHINI

Tese de Mestrado

CONTROLE DO COMPUTADOR USANDO MOVIMENTOS DO CORPO, IDENTIFICADOS POR UM ADESIVO, CAPTURADOS POR UMA CÂMERA DE VÍDEO WEBCAM

Banca Examinadora:
Prof. Dr. Armando Freitas da Rocha
FMUSP/USP

Prof. Dr. Roberto de Alencar Lotufo
DCA/FEEC/UNICAMP

Prof. Dr. Fernando José Von Zuben
DCA/FEEC/UNICAMP

Prof. Dr. Luís Fernandez Lopez
FMUSP/USP

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação (FEEC) da Universidade Estadual de Campinas (UNICAMP), como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica.

Área de Concentração: Engenharia de Computação

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

L962c

Lucchini, Fábio Luís Picelli
Controle do computador usando movimentos do corpo, identificados por um adesivo, capturados por uma câmera de vídeo WebCam / Fábio Luís Picelli Lucchini.--Campinas, SP: [s.n.], 2001.

Orientadores: Armando Freitas da Rocha, Fernando Antonio Campos Gomide.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Reconhecimento de padrões. 2. Visão por computador. 3. Informática para deficientes físicos. 4. Controle automático – Sensibilidade. 5. Interfaces – (Computador). 6. Processamento de imagens. I. Rocha, Armando Freitas da. II. Gomide Fernando Antonio Campos. III.Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Resumo

Neste trabalho, descreve-se a implementação de um sistema computacional, denominado REMMC (Reconhecimento Especial de Marcadores para Manuseio de Computadores), para realizar a tarefa de detecção e reconhecimento de um objeto inserido dentro de uma imagem. Essa imagem é recebida por uma câmera de vídeo webcam. Nesta imagem, o sistema procura um objeto específico previamente identificado com a finalidade de encontrar a sua posição e seguir o seu movimento dentro da imagem. A movimentação desse objeto, chamado objeto marcador, é transmitido ao sistema operacional para simular o movimento do mouse. Com essa alternativa, é possível controlar o computador sem colocar a mão no teclado ou no mouse. Esse objeto marcador pode ser um adesivo ou colante colocado em qualquer parte do corpo do usuário.
Esse sistema poderá ser utilizado por pessoas com deficiências motoras que tenham dificuldades ou impossibilidade de usar o mouse ou teclado do computador, detectando o melhor movimento que o indivíduo seja capaz de realizar para controlar o computador.
Desta forma, o sistema proposto é uma alternativa eficiente às soluções ad hoc baseadas no desenvolvimento de periféricos que são desenhados especificamente para cada deficiente. Essas soluções, além de serem mais caras, dependem sempre de que um técnico se disponibilize a construí-las.
O sistema se baseia no conhecimento fisiológico do sistema visual humano, utilizando técnicas de linguagens formais nebulosas e de visão computacional. A eficiência do sistema foi testada com adultos e crianças normais e também crianças portadoras de deficiências motoras leves e graves. Os resultados comprovam a possibilidade de utilização do sistema para controle de softwares, inclusive educacional, embora o treinamento requerido para tanto dependa da capacidade motora inicial do indivíduo. O sistema poderá, portanto, ser utilizado tanto para lazer como para facilitar o aprendizado, quer em sistemas locais ou em processos de educação à distância (Internet).

Abstract

A software was developed to track an adhesive target placed in the body segment having the best muscle control in motor disabled people, and to use the information about target displacement to simulate the computer mouse control or keyboard typing. A commercial webcam is used to monitor the target displacement. The software REMMC (Special for Computers) is aimed to be a general and low cost solution to aid disabled people to get access to the digital world, contrasting with the frequent ad hoc and high cost solutions designed to solve the problem of a specific subject.
The strategy used for image processing is inspired in the properties of the human visual system, and take profit from the theories of fuzzy formal language and vision computing.
The system was successfully tested with normal and disabled people, allowing subjects to operate commercial software such as Word®; ENSCER® - an educational software, etc. It is also possible to use the software for motor control training. It is assumed, therefore, that the software may allow access of disabled people to the digital world for both pleasure and educational purposes.

Agradecimentos

A Deus.

Aos meus pais, Olavo e Maria Albertina, pelo apoio, carinho e amor que sempre depositaram em mim.

Agradeço muito ao Professor Armando Freitas da Rocha pela orientação essencial ao desenvolvimento deste trabalho e pelo convívio enriquecedor no aspecto profissional e pessoal.

Ao Prof. Fernando Gomide pelas sugestões e comentários na fase final do nosso trabalho.

Ao pessoal do Departamento de Engenharia de Computação e Automação Industrial (DCA) da Faculdade de Engenharia Elétrica e de Computação (FEEC) pela gentileza com que sempre me atenderam.

A meus irmãos, Marcos, Roberto e Aline, que sempre estiveram comigo me ajudando diretamente ou indiretamente em toda a minha vida.

A meus familiares que incondicionalmente estiveram comigo em todos os momentos da minha vida, principalmente à minha avó, Elaine, que com todo seu carinho e amor me ajudou a suportar todas as dificuldades desta jornada.

À minha noiva Alessandra Campetela, pela paciência, compreensão, carinho, amor e por me trazer alegria nos momentos mais estressantes.

A meu amigo e colega Ednilson Cesar Rodella por sua amizade e pelas grandes contribuições realizadas a este trabalho no dia a dia. Sem sua valiosa ajuda não teria sido possível a realização com êxito deste trabalho.

Ao amigo Mateus Guilherme Fuini que sempre estive do meu lado me ajudando e dando força principalmente na fase de testes e avaliações do sistema.

A Cilene Campetela pela correção e revisão da dissertação

Aos meus amigos da Eina (Estudos em Inteligência Natural e Artificial), Márcia, Luciana, Samantha, Flávia, Cecília, Ana Paula, que sempre me ajudaram com suas sugestões e incentivos para a realização deste trabalho e também na realização de testes usando o sistema. Em especial à Cássia, pela confecção de alguns dos desenhos que fazem parte deste trabalho, e à Andrea, que desde o início me ajudou com ótimas sugestões e a paciência necessária para testar o software.

Ao amigo Marcos Paulo (Caco) que contribuiu muito no desenvolvimento deste trabalho, principalmente ajudando e dando dicas para a construção de algumas ferramentas usadas no software.

À Fapesp (Fundação de Amparo à Pesquisa do Estado de São Paulo) por seu suporte financeiro, através de uma bolsa de estudos, que permitiu que eu pudesse me dedicar integralmente ao desenvolvimento deste trabalho. Processo número 98/16433-1

A todos, os meus mais sinceros e
Profundos agradecimentos
MUITO OBRIGADO

Sumário

Lista de Figuras

Lista de Tabelas

1. Introdução

A adoção da postura bípede foi a solução encontrada pela natureza para a adaptação dos antecessores do *Homo Sapiens* às novas condições de vida, quando eles abandonaram as florestas em troca das savanas. Esta mudança de postura, enquanto solução para os problemas mais imediatos de sobrevivência, criou outras dificuldades para estes primeiros animais e todos os seus descendentes *(Aiello, 1997; Joseph; 1996; Orsntein, 1991*), tais como adaptação termodinâmica, metabólica e reprodução. A postura bípede foi acompanhada por um estreitamento da bacia, requerendo uma antecipação da liberação do feto. Este fato exigiu um aumento nos laços de integração familiar e social como solução para atendimento das necessidades desse cérebro que nasce imaturo. A imaturidade cerebral, entretanto, passa a ser um fator positivo, pois aumenta a capacidade adaptativa do cérebro ao ambiente no qual deve se desenvolver o homem. As dificuldades de liberação do feto, porém, continuam como fatores etiológicos importantes para inúmeras disfunções cerebrais, entre as quais, a paralisia cerebral (*Baron, Fennell and Voeller, 1995; Batchellor and Dean, 1996; Spreen, Risser, Edgell, 1955*;). Outras patologias congênitas, peri e pós-natais de origem genética, infecciosa ou dependente da saúde materna, completam as causas mais freqüentes da redução da capacidade cerebral, e principalmente dos distúrbios motores observados em crianças (*Baron-Cohen, 1995; Duane, 1995; DeFries and Gillis, 1995; Lubs et al, 1995*).

Um fator complicador no processo de educação da criança deficiente é a variabilidade das estruturas cerebrais afetadas, o que requer um processo de diagnóstico refinado e prático, que possibilite uma melhor compreensão das dificuldades da criança, fundamental para uma boa orientação do processo pedagógico. Por esses motivos, *Rocha et al, 1997* propuseram o desenvolvimento do sistema ENSCER®[1] como um sistema informatizado para a análise quantitativa do EEG (Eletroencefalografia); processamento e padronização do processo de diagnose médica sobre o cérebro deficiente e para geração de

[1] http://www.enscer.com.br/

um conjunto de jogos educacionais para complementar o processo de ensino do cérebro deficiente. A utilização desse sistema com diferentes grupos de crianças tem mostrado que o computador pode ajudar muito o desenvolvimento cognitivo de crianças deficientes cerebrais, inclusive portadores de paralisia cerebral (PC).

As crianças com PC podem apresentar diferentes graus de dificuldades motoras, associadas ou não a uma deficiência mental (*Aicardi, 1998*). Quando a deficiência motora é leve, o manuseio do computador pode ser feito com equipamento normal (teclado e mouse), porém quando a habilidade motora se reduz acentuadamente, a solução tem sido o desenvolvimento de mouses e teclados especiais (*Santarosa, 1994a-1994b*) ou dos chamados ponteiros, que devem ser fixados na criança por meio de cintas para permitir a utilização de movimentos da cabeça para manuseio do teclado (*Heidrich, 1999*). Entretanto, essas soluções são caras, pois devem ser, em geral, projetadas individualmente.

Existem diversos equipamentos eletrônicos e softwares comerciais que ajudam o deficiente físico na operação do computador, como exemplo o WinScan[1] (*Lacefield & Garthee, 1995*), que mostra um quadro de operações e o usuário de alguma forma seleciona a opção desejada para simular a ação, HeadMaster[2], da Prentke Romich, e o HeadMouse[3] da Words+, que convertem movimentos da cabeça do usuário para movimentos do cursor do mouse na tela através de um emissor de luz infravermelha e um sensor óptico. A operação de click do mouse deve ser feita por um outro dispositivo adicional, acionado por uma outra parte do corpo. Além desses existem outras soluções descritas por *Capovilla et al. (1998, 2000).*

A utilização de recursos tecnológicos na educação de indivíduos com necessidades especiais tem como meta, portanto, opor-se aos métodos mais tradicionais empregados na (re)educação e (re)habilitação destes indivíduos (*Mantoan e Valente, 1997; Valente, 1998*, *1999*). Neste sentido, não se trata de usar uma ferramenta tecnológica com o objetivo de "corrigir" uma "anormalidade intelectual" (física, sensorial, cognitiva), mas sim de oferecer

[1] http://www.acsw.com/
[2] http://www.prentrom.com/
[3] http://www.words-plus.com/

assistência às necessidades do indivíduo para que ele possa desenvolver o seu potencial cognitivo, criativo e humano. Afasta-se, assim, das propostas educacionais que se centram em métodos e técnicas desenvolvidas para corrigir ou minimizar tais desvios.

Da mesma forma, o presente trabalho se propõem a dar mais uma alternativa de software capaz de controlar o mouse através de movimentos de um marcador ou algum objeto previamente identificado pelo sistema. Esse marcador pode ser uma etiqueta adesiva colocada em alguma parte do corpo do usuário (no qual tenha o maior controle motor possível). Posicionamos uma câmera de vídeo focando o marcador em questão e transmitimos o seu movimento ao software que controlará o ponteiro do mouse no sistema operacional.

A grande maioria das soluções disponíveis comercialmente são caras ou não atendem a todos de uma maneira geral. A nossa proposta foi o desenvolvimento de um software que pudesse dar mais uma alternativa ao usuário portador de deficiência física, de maneira que o atendesse de uma maneira mais geral, simples e barata. O nosso software necessita apenas de uma câmera de vídeo, por exemplo WebCam, e um computador com o sistema operacional Windows®. Com esses itens, o usuário poderá fazer uso do sistema.

Além disso, o acesso ao computador ampliará as possibilidades de sucesso escolar ou profissional do deficiente motor, pois com o avanço tecnológico atual e as possibilidades do uso da Internet, geram enormes possibilidades de estudos e trabalhos avançados via computador. Assim, as barreiras da comunicação do indivíduo são atenuadas e suas idéias e opiniões podem ser levadas a qualquer parte do mundo.

1.1. Objetivos e Organização

O objetivo é desenvolver um sistema computacional, denominado REMMC (Reconhecimento Especial de Marcadores para Manuseio de Computadores), para realizar a tarefa de detecção e reconhecimento de um objeto inserido dentro de uma imagem. Essa imagem é recebida por uma câmera de vídeo webcam. Nela, o sistema procura um objeto específico previamente identificado com a finalidade de encontrar a sua posição e seguir o seu movimento dentro da imagem.

A movimentação desse objeto, chamado objeto marcador, é transmitido ao sistema operacional para controlar o movimento do mouse. Com essa alternativa é possível interagir com o computador sem colocar a mão no teclado ou no mouse. Esse objeto marcador pode ser um adesivo ou colante colocado em qualquer parte do corpo do usuário.

O sistema visa ser utilizado por pessoas portadoras de deficiência motora. Este tipo de *software* deve ser extremamente simples e de fácil ajuste às inúmeras condições de luminância, às restrições de movimentos, às deformações físicas *etc*. Para que esta solução se torne viável o custo de aquisição do equipamento está limitada ao valor de um computador pessoal mais uma câmera de vídeo webcam.

As técnicas utilizadas para o tratamento de imagens são propostas já adotadas na área de visão computacional, que apresentam diversos métodos de filtragem, segmentação e reconhecimento de padrões em imagens. Nesta pesquisa, as imagens utilizadas pelo sistema são capturadas através de uma câmera de vídeo conectada ao computador.

A escolha do ambiente *Windows*® no desenvolvimento do sistema se deve à ampla difusão deste sistema operacional entre os usuários, além de suportar uma grande variedade de dispositivos e *drivers* disponíveis no mercado.

Este trabalho está organizado da seguinte maneira.

No capitulo 2, descreve-se o funcionamento básico do *software* REMMC, mostrando seu ambiente e algumas características utilizadas pelo *software* como, por exemplo, calibração e aquisição da imagem.

No capítulo 3, explica-se todo o procedimento de aquisição, tratamento e reconhecimento da imagem utilizando bibliotecas desenvolvidas em Visual C++® e também como o sistema manipula o mouse através de informações obtidas da imagem.

No capítulo 4, mostra-se o ambiente de utilização do software e algumas características relacionadas com o marcador e como configurá-lo.

No capítulo 5, decompõe-se todo o procedimento realizado no software, que transforma uma seqüência de movimentos de um determinado objeto em controle do mouse.

No capítulo 6, faz-se um resumo de todo o processamento realizado.

No capítulo 7, ilustram-se os testes realizados em diversas condições, aplicados a várias crianças e adultos, a fim de comprovar a eficiência do software como uma ferramenta adequada a pessoas com deficiência motora ou física.

O capítulo 8, conclusivo, contém as considerações finais referentes ao desenvolvimento desta pesquisa.

2. Sistemas Sensoriais

2.1. Sistema Sensorial da Visão Humana

Neste capítulo, descrevemos um resumo dos conceitos fundamentais do sistema sensorial da visão humana, os quais são abordados de maneira simples e genérica com a finalidade de mostrar sucintamente as principais característica da visão humana. Com base nos conceitos da visão biológica, adaptamos a sua teoria em um sistema visual artificial, conforme trabalhos anteriores (*Rocha 1997,1999,2000; Alegre 1993a,1993b*).

2.1.1. Observando o mundo

Conforme diz *Rocha – (1999)* sobre os sistemas sensoriais visuais, o cérebro utiliza receptores distintos para medir diferentes tipos de variações energéticas, e deste modo coletar dados sobre a dinâmica do ambiente em que vive. Os dados sensoriais coletados pelos receptores são processados por um conjunto de neurônios que se distribuem por várias partes do cérebro. O conjunto de neurônios que se especializam para o processamento dos dados sensoriais, fornecidos por receptores que medem um mesmo tipo de energia, formam um sistema sensorial.

As ondas luminosas emitidas pelos objetos que vemos no mundo que nos cerca criam uma imagem deste objeto sobre a retina e estimulam dois tipos diferentes de receptores: os bastonetes e os cones. Os cones são sensíveis a três cores: verde, vermelho e azul, enquanto que os bastonetes podem ser ativados por ondas luminosas de qualquer freqüência.

Os cones geram três imagens com cores diferentes (coluna da esquerda na Fig. 2-1) da mesma cena (topo da coluna do meio na Fig. 2-1), enquanto os bastonetes fornecem ao cérebro uma imagem em tons de cinza (topo da terceira coluna na Fig. 2-1), pois medem a luminância da cena observada.

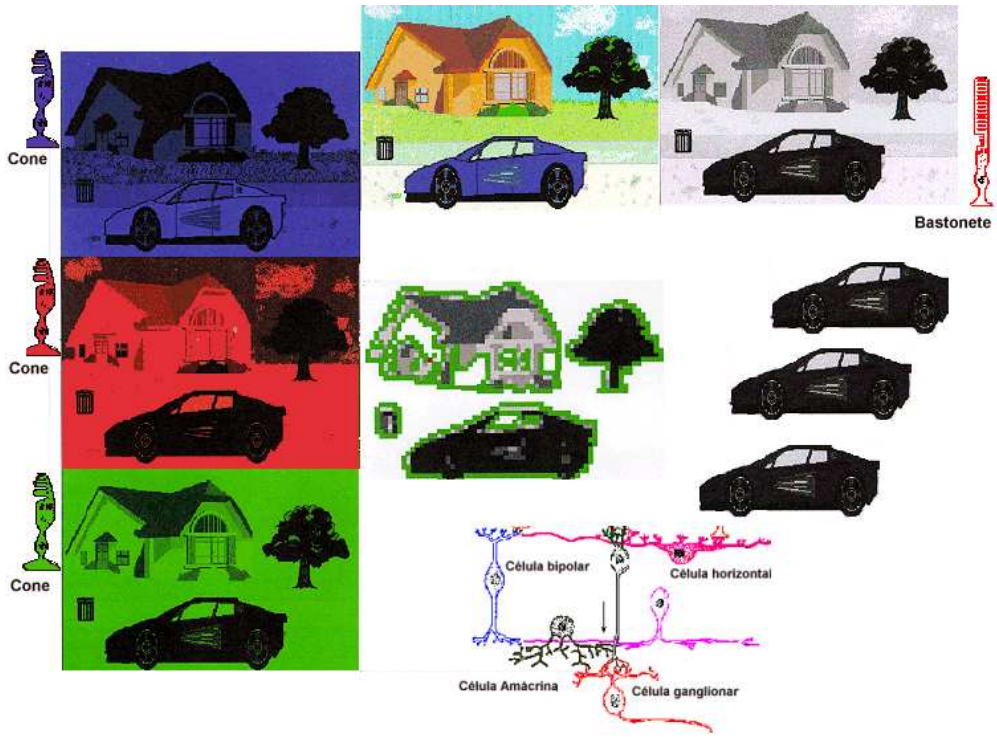


Fig. 2-1 - As distintas imagens da mesma cena geradas na retina

As diferentes células da retina se associam formando diferentes tipos de circuitos, que se especializam em processar certas características básicas da cena observada, como por exemplo contrastes de luminosidade (cena do meio da segunda coluna na Fig. 2-1), movimento de objetos (ilustrada pelo movimento do carro na coluna da direita na Fig. 2-1).

A retina realiza um processamento complexo da imagem visual e gera tipos distintos de informação para as outras áreas cerebrais:

a) três descrições nas cores básicas verde, vermelho e azul;
b) uma descrição da luminância da cena;
c) uma descrição dos contrastes de luminosidade; e
d) uma descrição dos movimentos dos elementos da cena.

O papel dos neurônios localizados em diversas áreas cerebrais é analisar características particulares em cada uma dessas diferentes imagens geradas pela retina e recombiná-las para reconstruir a cena original, da maneira com iremos finalmente enxergá-

la. Temos, então, um processo que inicialmente decompõe a cena e analisa características especiais da imagem, e depois remonta a realidade a partir desta análise primária.

2.1.2. Reconhecendo propriedades básicas das imagens

Inicialmente, as informações provenientes da retina ativam as áreas visuais primárias, localizadas no lobo occipital, cuja função é extrair características básicas da imagem, tais como direções predominantes de linhas; identificação de ângulos (Fig. 2-2); composição de cores (Fig. 2-3); análise de direções e velocidades de movimentação (Fig. 2-4), etc. O papel destas áreas no processamento visual é, portanto, decompor a imagem em um conjunto de características básicas, que serão utilizadas no reconhecimento dos elementos que compõem a cena a ser analisada.

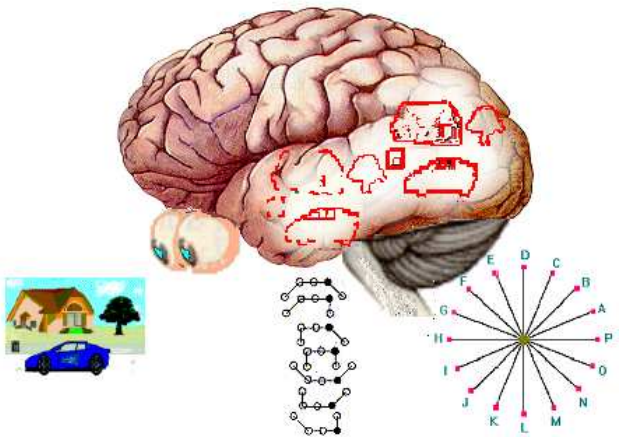


Fig. 2-2 - A análise cortical de linhas e ângulo

Na seqüência, o resultado do processamento na áreas visuais secundárias é transferido para áreas de associação, distribuídas nos lobos temporal e parietal. O papel destas áreas de associação é reconhecer os principais elementos que compõem a cena, a partir de suas características básicas. Um quadrado é reconhecido por ser composto por linhas de tamanho semelhante e que formam ângulos de aproximadamente 90°. Uma casa será identificada por ser composta por linhas de direções específicas, que formam ângulos característicos, etc. Cada elemento é, portanto, reconhecido pelas relações específicas partilhadas por um conjunto de características definidas. Quando se aprende a reconhecer

um objeto, se aprende a identificar este conjunto de características específicas e as relações estabelecidas entre elas.

Enquanto os objetos principais de uma cena são reconhecidos pelas suas formas, outras áreas visuais fazem uma análise da distribuição das cores nos campo visuais (Fig. 2-1) ou dos movimentos aí registrados (Fig. 2-4). Estas análises ocorrem em paralelo e dissociadas das análises de formas. A unificação de todas estas informações para recompor a cena observada é tarefa de áreas de associação secundárias, localizadas, principalmente no córtex parietal (Fig. 2-1).

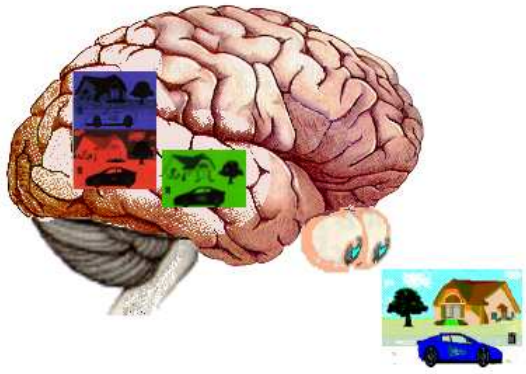


Fig. 2-3 - A análise cortical de cores

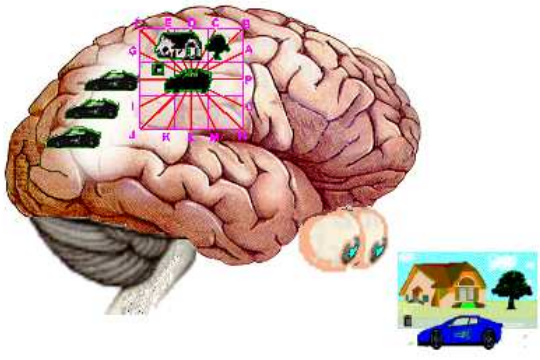


Fig. 2-4 - A análise cortical do movimento e localização espacial

2.1.3. Recompondo a cena

A segunda etapa do processamento sensorial envolve a recomposição da cena final a partir dos elementos individualmente reconhecidos, e das relações partilhadas por eles na imagem observada e identificadas pelo cérebro.

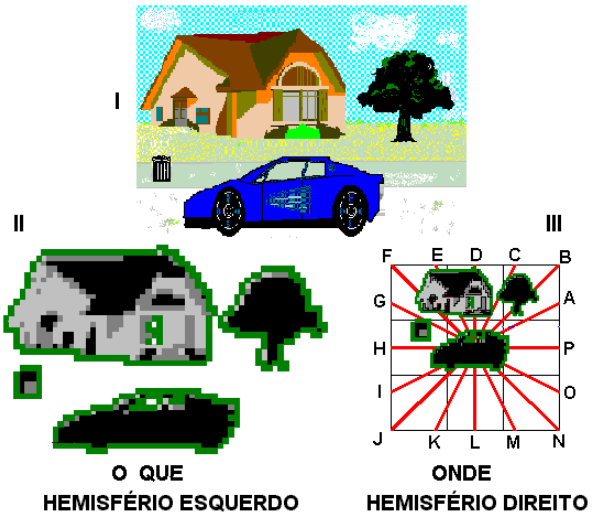


Fig. 2-5 - A especialização hemisférica

É postulado que a identificação dos objetos é um trabalho preferencial do hemisfério esquerdo, enquanto que a análise das relações espaciais fica mais a cargo do hemisfério cerebral direito (Fig. 2-5). Esta análise espacial envolve também a composição e análise da coerência dos movimentos associados a cada um dos objetos que compõem a cena visualizada.

Supõe-se, assim, que o hemisfério direito tem um papel muito importante no processamento visual, uma vez que a recomposição espacial das relações partilhadas pelos objetos da cena é um processamento muito importante para a criação da imagem visualizada da cena original.

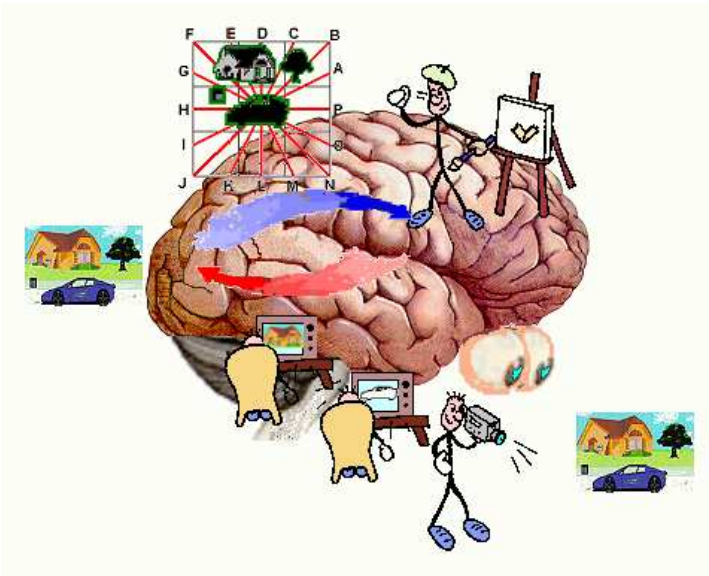


Fig. 2-6 - A ativação e a retroativação durante o processamento visual

Deve-se ressaltar que durante a primeira etapa do processamento sensorial, a ativação cerebral obedece um gradiente posteroanterior (seta azul na Fig. 2-6). Primeiro são ativadas as áreas occipitais responsáveis pela análise das características elementares (linhas, ângulos, cores, etc.) dos elementos que compõem a cena.

Na segunda etapa do processamento sensorial, quando a cena final é recomposta a partir dos seus elementos identificados, há uma reversão da ordem de ativação das áreas corticais (seta vermelha na Fig. 2-6). Agora, serão as áreas occipitais as últimas a serem reativadas pelos neurônios que participam do reconhecimento dos componentes e suas relações na cena. Acredita-se que a recomposição perceptual da cena observada ocorra a partir desta reativação secundária das áreas visuais primárias.

2.2. *Um Sistema Sensorial Formal*

Conforme proposto por *Rocha (1997; 1999) e Serapião (1996; 1997)*, a teoria de linguagens formais nebulosas pode ser utilizada para simular o funcionamento do sistema visual humano. O processamento no nível da retina pode ser simulado por vários tipos de filtros e algoritmos para extração de contorno, enquanto que filtros de segmentação podem simular circuitos pré-corticais (Fig. 2-7).

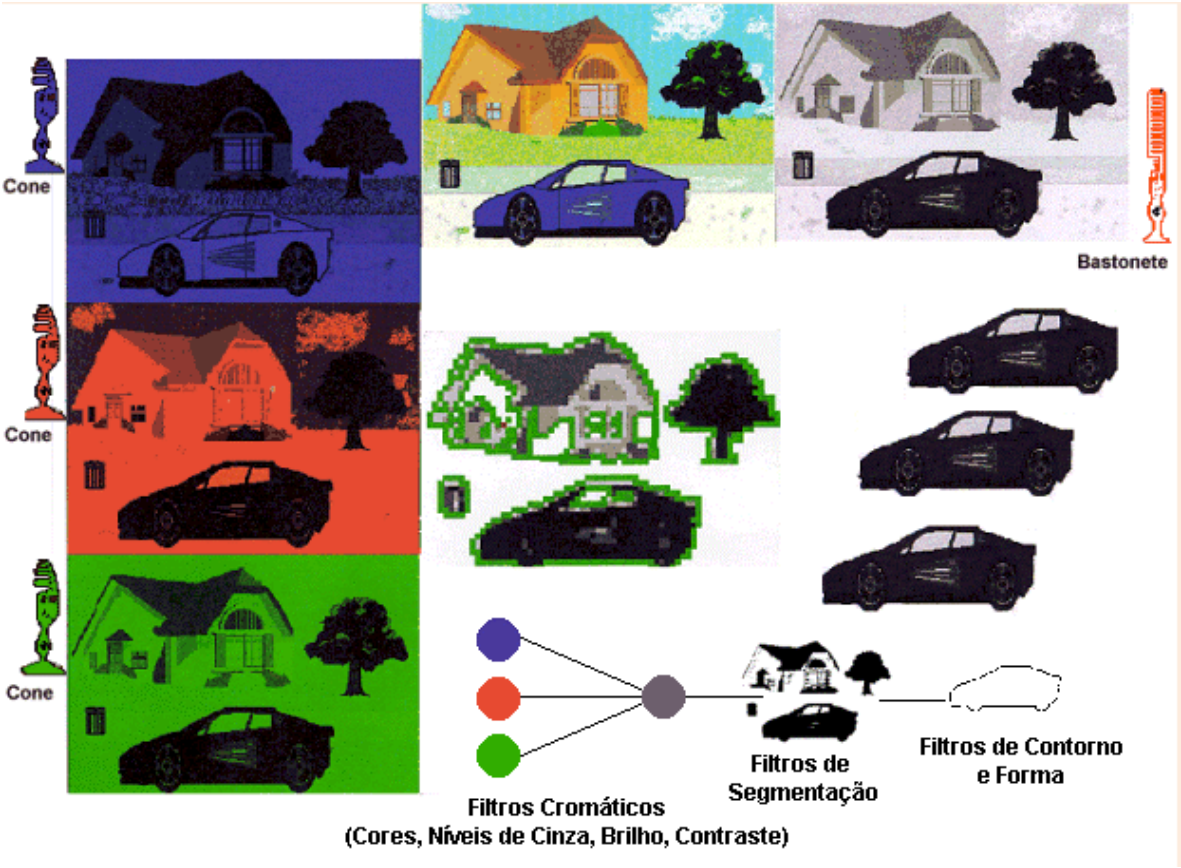


Fig. 2-7 - Organização computacional dos filtros

Podemos dizer então que uma imagem refere-se a uma função de intensidade luminosa bidimensional, denotada por *f(x,y)*, em que o valor ou amplitude de *f* nas coordenadas espaciais *(x,y)* dá a intensidade (brilho) da imagem naquele ponto. Como a luz é uma forma de energia , *f(x,y)* deve ser positiva e finita, isto é

$$\mathbf{0 < f(x, y) < \infty}$$

As imagens que as pessoas percebem em atividades visuais corriqueiras consistem de luz refletida dos objetos. A natureza básica de *f(x,y)* pode ser caracterizada por dois componentes: (1) a quantidade de luz incidida na cena sendo observada e (2) a quantidade de luz refletida pelo objetos na cena. Apropriadamente esses componentes são chamados de iluminação e reflectância, respectivamente, e são representados por *i(x,y)* e *r(x,y)*. O produto das funções *i(x,y)* e *r(x,y)* resulta *f(x,y)*.

$$\mathbf{f(x, y) = i(x,y)\, r(x,y)}$$

onde $\mathbf{0 < i(x,y) < \infty}$

e $\mathbf{0 < r(x,y) < 1}$

A equação acima indica que a reflectância (r) é limitada entre 0 (absorção total) e 1 (reflectância total). A natureza de *i(x,y)* é determinada pela fonte de luz, e *r(x,y)* é determinada pelas características dos objetos na cena.

Denominamos a intensidade de uma imagem monocromática *f* nas coordenadas *(x,y)* de nível de cinza *(l)* da imagem naquele ponto. Como mostra as equações acima, fica evidente que *l(x,y)* fica restrito ao intervalo.

$$L_{min} \leq l(x,y) \leq L_{max}$$

Em teoria, a única restrição sobre L_{min} é que seja um valor positivo e sobre L_{max} é que seja positivo, finito e maior que L_{min}.

O intervalo [L_{min} , L_{max}] é denominado escala de cinza. A prática comum é deslocar este intervalo para [0 , L], onde $l(x,y) = 0$ é considerado negro e $l(x,y) = L$ é considerado

branco. Todos os valores intermediários são tons de cinza variando continuamente entre o branco e o negro.

Supondo que seja dado um dispositivo de sensoriamento S para observar um conjunto de objetos O_i, pode-se definir a imagem I_i gerada por S para a observação do objeto O_i como:

$$I_i = \{ x_i , y_i , m_i \}_{\, i = 1 \text{ to } k}$$

onde m_i é o valor da medida numérica realizada por S na posição (x_i , y_i) sobre O_i, variando de 1 até k pixels do objeto. O conjunto de posições que podem ser sensoriadas por S definem um reticulado L. O sistema visual mostrado na Fig. 2-8, é composto de um dispositivo Lum para medida da luminância sobre L_1. Esse sistema pode ser utilizado para gerar imagens visuais I_i de objetos O_i focados sobre L_1, isto é

$$I_i = \{ x_i , y_i , l_i \}_{\, i = 1 \text{ to } k}$$

onde l_i é a medida de luminância realizada na posição (x_i , y_i), variando de 1 até k pixels.

Suponha um esquema como o *Sensor* (*Serapião et al. 1996*), que é um sistema hierárquico de sensoriamento inteligente que utiliza agentes primários para o reconhecimento de características elementares de objetos sensoriados por um dispositivo visual V, e outros agentes de maior grau hierárquico para o reconhecimento progressivo de características mais complexas dos objetos O_i observados por V. Uma das características dos agentes do sistema *Sensor* é sua capacidade de utilizar linguagens formais definidas por gramáticas G_j para recodificar as informações numéricas contidas em I_i. Essas gramáticas são definidas como:

$$\mathbf{G_j = \{ V_s, V_n, V_t, P, \eta)}$$

onde:

a) $\mathbf{V_s}$: é um conjunto de símbolos iniciais;
b) $\mathbf{V_n}$: é um conjunto de símbolos não-terminais
c) $\mathbf{V_t}$: é um conjunto de símbolos terminais;
d) η : é o elemento vazio, e
e) **P**: é o conjunto de regras **p** de reescrita definidas como

$$\mathbf{p : \alpha\, s_i\, \beta \rightarrow \alpha\, s_j\, \beta}$$
$$\mathbf{\alpha\,,\ \beta\,,\ s_j \in V_s \cup V_n \cup V_t \cup \eta \ e\ s_i \in V_s \cup V_n}$$

Em outras palavras, $\mathbf{p \in P}$ reescreve a cadeia $\mathbf{s_i}$ como a cadeia $\mathbf{s_j}$. $\mathbf{s_i}$ é definida como uma cadeia de símbolos de $\mathbf{V_s \cup V_n}$, que é a união do conjunto de símbolos iniciais e símbolos não-terminais. $\mathbf{s_j}$ é definida como uma cadeia de símbolos da união de todos os conjuntos de símbolos, que é $\mathbf{V_s \cup V_n \cup V_t \cup \eta}$. α e β denotam cadeias contextuais, isto é, $\mathbf{s_i}$ é reescrito em $\mathbf{s_j}$ no contexto definido por α e β. Para simplificar, atribui-se,

$$\mathbf{V^+ = V_S \cup V_n \ e}$$
$$\mathbf{V^* = V_s \cup V_n \cup V_t}$$

A seqüência de derivação $\mathbf{d(s_o,s_m)}$ da cadeia $\mathbf{s_m \in V^*}$de uma gramática **G** é o conjunto ordenado de reescrições requeridas para transformar o símbolo inicial $\mathbf{s_o \in V_s}$ em $\mathbf{s_m}$. Em outras palavras,

$$\mathbf{d(s_o,s_m) = \alpha\, s_o\, \beta \rightarrow \alpha\, s_1\, \beta \rightarrow ..... \alpha\, s_k\, \beta ..... \rightarrow \alpha\, s_m\, \beta}$$

Uma linguagem formal **L** é definida como um subconjunto de cadeias geradas por uma gramática **G.** As cadeias geradas por **G** e aceitas como pertencentes à **L** são chamadas fórmulas bem formadas (**fbf**) de **L** de acordo com **G**. A cadeia $\mathbf{s_j}$ produzida por **G** é uma **fbf** de **L** se ela pertencer a $\mathbf{V_t}$. Em outras palavras, a cadeia $\mathbf{s_m}$ aceita pela linguagem **L(G)** suportada por **G** são aquelas $\mathbf{fbf(s_o,s_m)}$ obtidas em

$$\mathbf{fbf(s_o,s_m) = d(s_o,s_m) = s_o \rightarrow \alpha\, s_1\, \beta \rightarrow ...\ \alpha\, s_k\, \beta ..... \rightarrow s_m\ ,\ s_m \in V_t}$$

Assim, a sentença $\alpha\ \mathbf{s_o}\ \beta$ é aceita como uma sentença de **L** somente se existir pelo menos uma $\mathbf{fbf(\alpha\ s_o\ \beta, s_m)}$ para reescrevê-la em $\mathbf{s_m}$ pertencente à $\mathbf{V_t}$.

O processamento de qualquer sentença da seqüência de derivação $\mathbf{d(s_o,s_m)}$ é um conjunto seqüencialmente ordenado de operações de reescrita, cada uma envolvendo os seguintes passos:

1) **Comparação**: o lado esquerdo da provável regra de reescrita é comparado com os símbolos da cadeia $\mathbf{s_i}$ a ser processada. Se esta comparação for bem sucedida, então

2) **Reescrita:** a subcadeia selecionada em $\mathbf{s_i}$ é substituída pelo lado direito da regra de rescrita aceita. Finalmente;

3) **Aceitação:** a pertinência da cadeia derivada $\mathbf{s_j}$ a $\mathbf{V_t}$ é calculada como uma medida definida no intervalo fechado **[0,1].**

No exemplo da Fig. 2-8, o agente primário Cont do *Sensor* recodifica a imagem I_i gerada por V sobre um objeto O_i em uma cadeia S_1 de caracteres $c^{(j)}$ pertencentes a um dicionário C_1 { p = preto , b = branco }e suportada por uma gramática G_1, isto é:

$$S_1 = c^{(n)}$$

tal que

$$c^{(j)} = p \text{ se } l_i < \alpha, \text{ caso contrário } c^{(j)} = b, \text{ na posição } (x_i , y_i)$$

onde $c^{(j)}$ é o jésimo caractere de S_1 e n é igual ao número de posições de sensoriamento em L_1.

O agente Bor na Fig. 2-8, rescreve a Sentença S_1 em outra cadeia S_2 suportada por uma gramática G_2 definida sobre o seguinte dicionário C_2 = { n, f } de acordo com as regras

$$pb \rightarrow n\ ,\ bp \rightarrow n\ ,\ bb \rightarrow f\ ,\ pp \rightarrow f$$

Em outras palavras, o agente Bor gera uma sentença S_2 que descreve a borda do objeto O_i sensoriado por V, sempre que S_2 contiver uma subcadeia

$$B_2 = c_2^k \ldots c_2^m$$

tal que qualquer $c_2^k = n$ para qualquer $c_2^k \in B_2$ e **k** e **m** estão associados a pontos vizinhos em L_1.

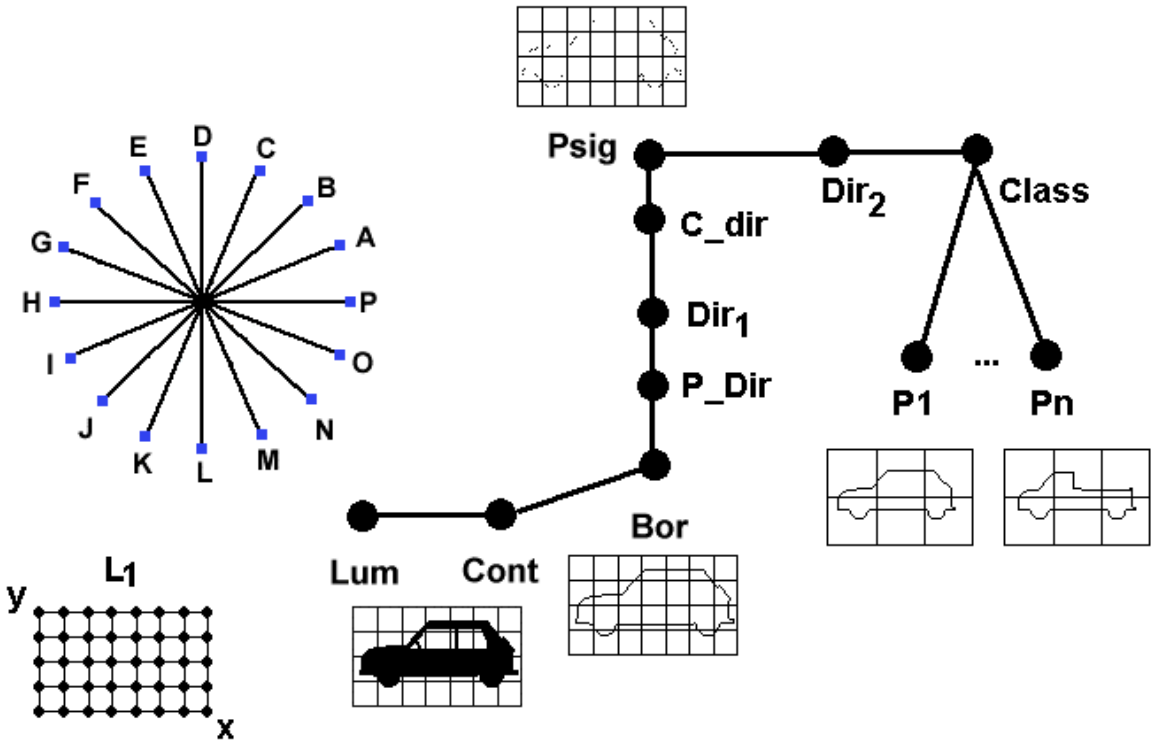


Fig. 2-8 - Um sistema de inteligência artificial distribuído composto por agentes especializados para o reconhecimento de elementos, analisando suas características e a complexidade de seus modelos de uma imagem. Estes agentes são desenvolvidos para descrever o comportamento dos neurônios no caso do sistema visual natural. Também a hierarquia imposta aos agentes é colocada como a organização das áreas do sistema visual natural.

A distância $d_{i,j}$ entre duas posições l_i , l_j em L_1 pode ser definida como

$$d_{i,j} = \{ d_x , d_y \}$$

$$d_x = x_i - x_j, d_y = y_i - y_j$$

O agente P_Dir na Fig. 2-8 gera uma sentença S_3 que descreve as direções entre dois caracteres consecutivos de uma borda B_2, utilizando o dicionário C_3

C_3 = { p = positivo, n = negativo, z = zero, a = vazio, b = pequeno, c = médio, d = grande, e = enorme }

tal que

Negativo	dx ou dy < 0
Positivo	dx ou dy > 0
Zero	dx ou dy = 0
Nulo	$dx / dy < \alpha_1$
Pequeno	$\alpha_1 \leq dx / dy < \alpha_2$
Médio	$\alpha_2 \leq dx / dy < \alpha_3$
Grande	$\alpha_3 \leq dx / dy < \alpha_4$
Enorme	$dx / dy \geq \alpha_4$

onde S_3 é uma sentença suportada por uma gramática G_3

$$S_3 = \{ c_{x1} c_{y1} c_2 \}^j$$

tal que

$$c_{x1} , c_{y1} \in \{ p, n, z \}, c_2 \in \{ b, c, d, e \}$$

Em outras palavras, S_3 descreve as direções entre elementos consecutivos da borda B_2 através de uma gramática onde C_3 que classifica um ângulo e c_{x1} c_{y1} identificam o seu quadrante.

O agente Dir_1 reescreve S_3 em uma outra cadeia S_4 suportada pela gramática G_4 utilizando o seguinte dicionário C_4 = { A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P } de acordo com as seguintes regras

$$\begin{Bmatrix} \mathbf{p*a \rightarrow P} & \mathbf{ppb \rightarrow A} & \mathbf{ppc \rightarrow B} & \mathbf{ppd \rightarrow C} \\ \mathbf{*pe \rightarrow D} & \mathbf{npd \rightarrow E} & \mathbf{npc \rightarrow F} & \mathbf{npb \rightarrow G} \\ \mathbf{n*a \rightarrow H} & \mathbf{nnb \rightarrow I} & \mathbf{nnc \rightarrow J} & \mathbf{nnd \rightarrow K} \\ \mathbf{*ne \rightarrow L} & \mathbf{pnd \rightarrow M} & \mathbf{pnc \rightarrow N} & \mathbf{pnb \rightarrow O} \end{Bmatrix}$$

onde * significa “qualquer direção”.

Por exemplo, a cadeia gerada pelo agente Dir_1, que descreve o contorno do carro na Fig. 2-8:

HDDCCPCPPPPPCCCCCPPPPPPPPPPPMPMCMLLKKHEEEHHHHHHHHHHHHHHHHKKKEHEHHHD

Na seqüência, o agente C_Dir rescreve a cadeia S_4 em uma cadeia S_5 composta por símbolos deste dicionário:

C_5 = { z = Sem Mudança , c = Mudança no Sentido Anti-horário , d = Mudança no Sentido Horário }

descrevendo as mudanças de direções entre os pontos da figura, associadas com dois símbolos consecutivos de S_4. Assim, S_5 descreve as mudanças de direções da borda da figura do carro.

O agente Psig pesquisa a existência das subcadeias: zcc, dcc, zdd, cdd, czc, dzd, zzc, zzd, zcz, e zdz, em S_5 para gerar um vetor v_6 contendo coordenadas *xy* da localização das subcadeias. Este vetor descreve os pontos da borda Psig (pontos significativos) onde as mudanças de direções ocorreram. O agente Psig usa a seguinte gramática G_5 sobre o dicionário C_5, para gerar, uma cadeia S_6 de acordo com as seguintes regras:

$$\begin{Bmatrix} \mathbf{zcc \rightarrow S} & \mathbf{dcc \rightarrow S} & \mathbf{zdd \rightarrow S} & \mathbf{cdd \rightarrow S} \\ \mathbf{czc \rightarrow S} & \mathbf{dzd \rightarrow S} & \mathbf{zzc \rightarrow S} & \mathbf{zzd \rightarrow S} \\ \mathbf{zcz \rightarrow S} & \mathbf{zdz \rightarrow S} & & \end{Bmatrix}$$

Senão → N

produzindo uma cadeia do tipo

$$S_6 = \{ \text{SNN.....SN} \}$$

Agora o agente Dir_2 (Fig. 2-8) recodifica S_6 em uma cadeia S_7 descrevendo as principais direções dentre os pontos Psig da borda B_2. No exemplo abaixo somente os pontos significativos do objeto carro da Fig. 2-8.

$$S_7 = \{ \text{HDDBBPCCPPMMCMLKKFEHHJKFFHD} \}$$

A partir da informação gerada por S_7, podemos agora armazenar esta informação como sendo um modelo de características de um objeto carro que é inserido em uma base de dados de conhecimento.

Com está definição anteriormente descrita, novas imagens podem ser classificadas e seus objetos podem ser confrontados com a base de dados de conhecimento, a fim de retirar informações necessárias para a identificação de objetos.

3. Funcionamento do Software REMMC

O software REMMC (Reconhecimento Especial de Marcadores para Manuseio de Computadores) foi desenvolvido com a finalidade de auxiliar o usuário na manipulação do computador sem a ajuda do mouse ou teclado. Neste capítulo, descrevemos as principais fases de uso e calibração do software, para deixá-lo operacional para qualquer indivíduo.

Como qualquer sistema de controle, o REMMC precisa ser calibrado, isto é, o sistema precisa conhecer a forma do marcador antes de iniciar o seu reconhecimento. Além disso, o software necessita de algumas características do ambiente em que o usuário está operando. Com essas informações, o sistema pode identificar e seguir o movimento do marcador e, em seguida, transformar esses movimentos em sinais de controle que serão transmitidos ao sistema operacional.

As principais etapas desse processo são resumidas na figura 2.1

Fig. 3-1 - Etapas da Execução do REMMC

3.1. Calibração

A calibração consiste em uma configuração inicial indicando ao software as características do ambiente quanto à luminosidade, ao formato e à cor do objeto marcador. Essas informações, depois de selecionadas, são registradas em um arquivo para que não haja mais a necessidade de sempre ajustar o sistema para a sua utilização.

Nesta etapa, o usuário deve informar qual é o marcador utilizado, porque o sistema precisa analisar e armazenar as características deste marcador para que ele possa ser reconhecido na fase de execução.

3.2. Aquisição da Imagem

Existe uma grande variedade de câmeras de vídeo no mercado, algumas com custo elevado e outras com custo bem acessível à maioria dos usuários. O foco principal do projeto foi a resolução do problema utilizando equipamentos de baixo custo. Isso só foi possível por dois motivos: a) o barateamento dos computadores, que possibilita, hoje em dia, que boa parte dos usuários tenha em casa um micro com capacidade de processamento igual ou superior a de um Pentium 333 Mhz com 64 Mb de RAM; b) a implementação dos filtros em C++, obtendo grande desempenho de velocidade na execução do software.

A obtenção da imagem é feita através de funções do tipo callback, que substituem as funções do driver da câmera por funções da própria aplicação do sistema, que recebem diretamente da memória os dados da imagem.

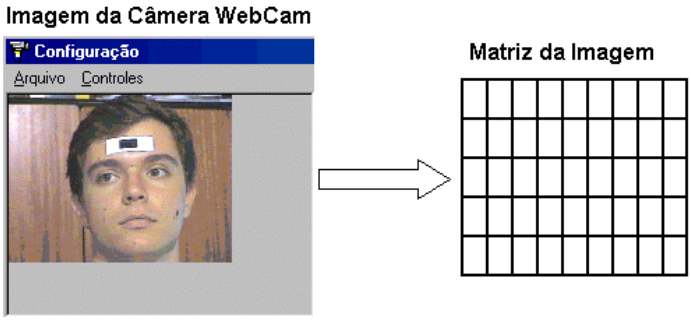


Fig. 3-2 - Aquisição da Imagem

3.3. Filtragem

O processamento se concentra na filtragem e separação de todos os objetos da imagem e a seleção do objeto marcador dentre os demais objetos. Para ser realizado tal processamento, é necessário que a imagem obtida passe por diversos filtros, até o momento em que a imagem original se transforme em uma imagem binária, mas mantendo as características básicas dos objetos da imagem original. Esses filtros estão agrupados na biblioteca Camera.dll, feita em Visual C++ 6.0®, que será descrita mais adiante.

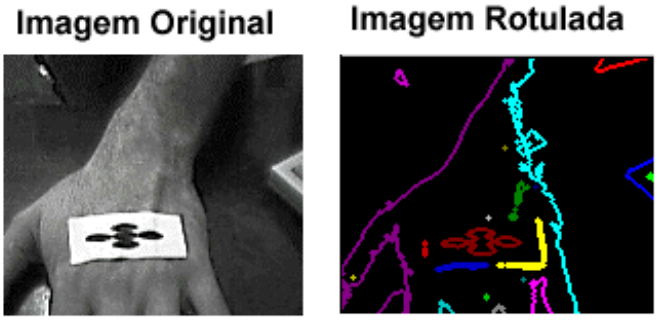


Fig. 3-3 – Filtragem

3.4. Encontrar o Marcador

Esta é a parte do processamento que vai classificar os objetos a partir de informações encontradas na imagem, geralmente de acordo com um banco de dados previamente estabelecido. Esta fase é geralmente aplicada após uma fase de segmentação da imagem e de uma fase de parametrização. A fase de parametrização identifica e calcula alguns parâmetros (pré-determinados) nos objetos segmentados (como exemplo de algum parâmetro, podemos citar o perímetro ou a área de uma determinada forma).

A classificação de um objeto chamado de marcador é realizada na fase de calibração. Quando o sistema está em execução, para cada imagem recebida da câmera será realizada uma pesquisa com finalidade de encontrar o marcador pré-identificado para rastrear o seu movimento. Significa que não basta encontrar o marcador na imagem, é preciso ainda identificar o seu posicionamento atual e informar para onde ele está se movendo.

Como mostra o exemplo da Fig. 3-4, faz-se a análise da imagem original retornando ao sistema a posição atual do marcador, utilizando um quadro de reconhecimento.

Fig. 3-4 - Identificando o Marcador

3.5. Tratar o Descolamento do Marcador

Se o sistema estiver apto a reconhecer o marcador dentro de uma imagem captada pela câmera, basta o usuário movimentar o marcador para a direção desejada e o próprio software informará ao sistema operacional o novo posicionamento do ponteiro do mouse. Do mesmo modo, existem movimentos padronizados para reproduzir o click ou duplo click do mouse.

Conforme o exemplo da Fig. 3-4, o objeto marcador foi encontrado no canto inferior direito do quadro. O programa tem que transferir esse movimento para o mouse, na direção diagonal direita inferior.

Neste caso, o mouse irá se movimentar nesta direção enquanto o marcador estiver nesta posição. Para que o mouse pare, basta retornar o marcador ao centro do quadro. Para chegar a esse resultado, o processo de transformação do marcador em movimento passa por duas fases:

3.5.1. Codificação do Movimento

A movimentação será detectada através da codificação da mudança relativa da posição do objeto marcador entre os frames *i* e *i+1* de acordo com a metodologia introduzida por *Serapião, (1996)* e ilustrada na Fig. 3-5. A posição no frame *i* será tomada como imagem origem de referência, e a posição do frame *i+1* será referenciada à direção desejada. Desta maneira, podemos comparar a trajetória do marcador e saber a direção do seu movimento. Neste processo, representamos o marcador pelo seu ponto central (centróide) para facilitar todas as comparações. Com isso utilizamos a posição do ponto central do marcador junto ao quadro de direções para identificar a sua posição atual. Desta forma, o sistema ganha simplicidade e rapidez.

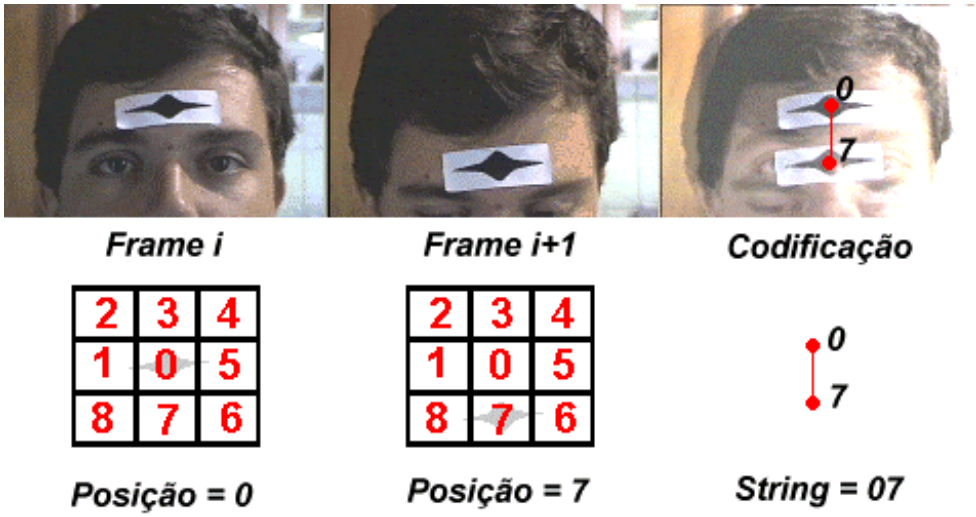


Fig. 3-5 - Codificando o Movimento do Marcador

Sabendo onde o marcador se localiza, é possível agora rastrear toda a trajetória do objeto marcador dentro da imagem, criando uma cadeia codificada de movimentos como mostra o exemplo abaixo:

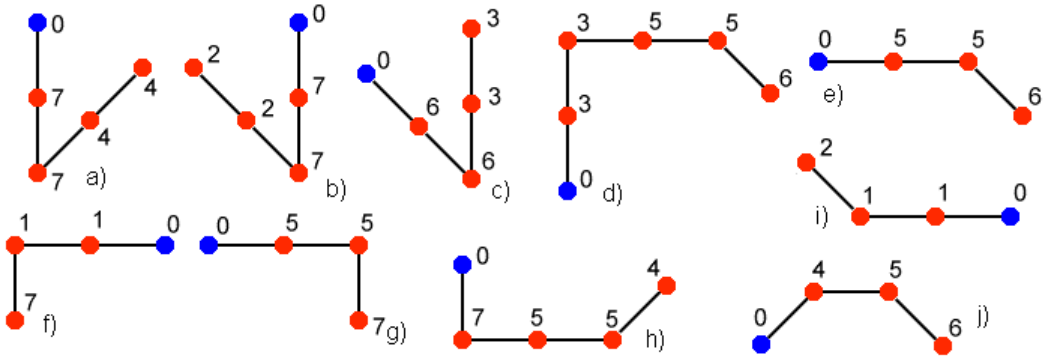


Fig. 3-6 - Exemplos de Movimentos Codificados

O ponto azul indica que o marcador está parado no centro, esta informação é representada como *0*. Quando se realiza um movimento, o sistema irá representar a sua trajetória conforme o quadro mostrado na Fig. 3-5.

Por exemplo, o item (*g*) tem uma cadeia codificada na seguinte forma: *0557*.

- *0*, Marcador no centro da imagem
- *5*, Marcador foi para o lado direito
- *5*, Marcador continua no lado direito
- *7*, Marcador foi para baixo

3.5.2. Interpretação da Cadeia Codificada do Movimento

Na sintaxe descrita no item anterior, a codificação do movimento gera uma cadeia de caracteres. O próximo passo é interpretar esta informação codificada, de forma que seja possível retirar operações de controle e filtrar prováveis ruídos.

No exemplo da Fig. 3-7, existem duas cadeias de caracteres que são representadas pela codificação do sistema REMMC. Nas cadeias abaixo o caractere *0*, significa que o mouse está parado.

Quando existir qualquer movimento diferente de *0,* o sistema inicializa um contador e espera a repetição do mesmo movimento até que atinja um determinado limite. Este limite é configurado na janela principal do sistema. Esta opção está relacionada com a velocidade de resposta do sistema para a movimentação do mouse. Se uma pessoa for mais rápida, pode reduzir este valor limite, ou aumentar se a deficiência ou a dificuldade física for muito acentuada.

Quando o limite de repetição for ultrapassado, o mouse começará a se movimentar na direção desejada. Quando o usuário quiser mudar de direção, terá que esperar novamente este ciclo, esta repetição do mesmo movimento é muito importante no sistema, porque ajuda a identificar ruídos ou operações de controle, como mostra a Fig. 3-7. Neste exemplo o limite está definido como sendo igual a 5.

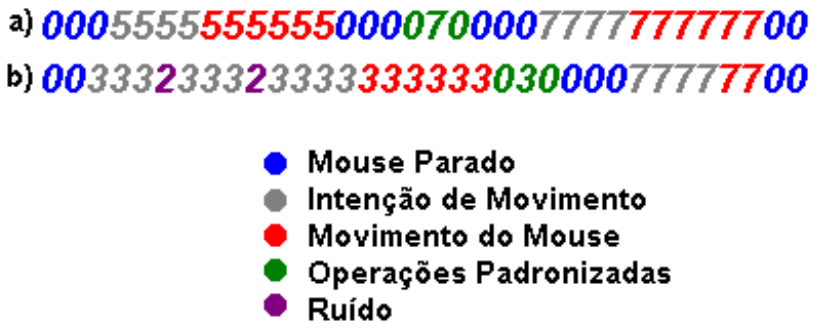


Fig. 3-7 – Cadeia Codificada de Caracteres

A cadeia de caracteres pode ser redefinida em partes menores, chamadas de *palavras*, que são delimitadas pelo caractere *0*, representativo do mouse parado no centro da imagem.

Pode-se definir três classes de palavras para controle de periféricos do computador, de acordo com seu número de caracteres:

- Curtas: $0\ C_1...C_i\ 0$
- Médias: $0\ C_1...C_j\ 0$ onde j>i
- Longas: $0\ C_1...C_n\ 0$ onde n>j

Onde : C: representa um movimento de 1 até 8
i: representa um limite de repetição curto
j: representa um limite de repetição médio
n: acima deste limite, representa um movimento do mouse

Utilizam-se:

As palavras curtas para implementar códigos de controle

- 030 (Click do Mouse)
- 070 (Duplo Click do Mouse)
- 010 (BackSpace)

As palavras médias para implementar o controle de teclado

- 0760 (Letra A)
- 0780 (Letra B)
- 0320 (Tecla ENTER)
- 0340 (Tecla ESC)

As palavras longas para controle de direção do mouse

- 05555555555550 (Mover Mouse para Direita)
- 07777777777770 (Mover Mouse para Baixo)

As operações padronizadas podem ser criadas a partir de movimentos que o usuário possa repetir de uma maneira simples e seqüencial, a fim de que o mesmo possa usufruir de todos os recursos do computador. No exemplo da Fig. 3-7, item (*a*), existe um movimento representado por *070*, que significa: *marcador no centro*, *marcador para baixo* e *volta o marcador para o centro*. O sistema REMMC irá interpretar essa informação como sendo um duplo click.

Quando percebe que o marcador não está mais no centro, o sistema espera um ciclo de repetição para começar a mover o mouse. Enquanto isso não ocorre, o sistema pode interpretar esta parte da informação, chamada de *palavra*, como sendo um sinal de controle.

Como o limite da repetição é configurável, o usuário não precisa ser tão rápido para dar o duplo click, ele pode realizar a mesma operação executando um movimento mais lento, por exemplo: *07770*. Este limite de repetição define os conjuntos de palavras curtas, médias ou longas, como discutido anteriormente.

No item *b* da Fig. 3-7, acontece o mesmo, porém o movimento padronizado *030,* representa o click do mouse. Outra característica importante são os possíveis ruídos gerados. Nesse caso, o sistema irá interpretá-lo, primeiramente, como sendo um movimento padronizado. Caso negativo, irá ignorá-lo e zerar o contador de intenção de movimento, reiniciando a interpretação normal do movimento. Este mesmo processo ocorre quando existe um movimento padronizado.

Tendo em vista estas opções, fica clara a possibilidade de ampliar os movimentos padronizados, a fim de que se possa abranger todas as teclas do teclado e todos os movimentos e controles do mouse, como mostra a Fig. 3-8:

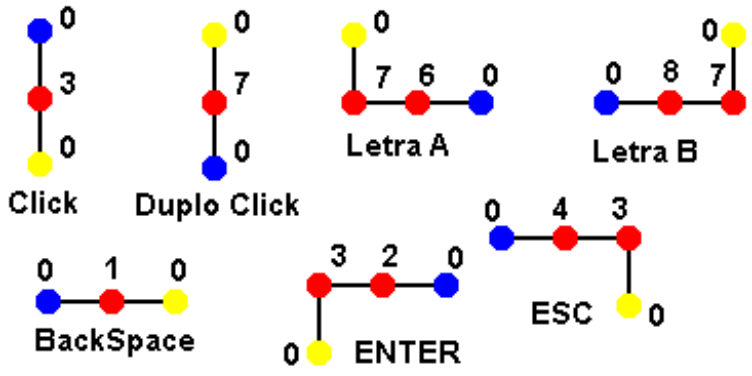


Fig. 3-8 - Palavras Codificadas no Sistema

No exemplo acima, os pontos amarelos simbolizam o início do movimento padronizado, o ponto azul indica o final da seqüência. Todos os caracteres entre estes dois

delimitadores representam a simulação da operação pretendida. Neste trabalho, realizamos somente testes com a utilização do mouse.

As opções aqui descritas podem ser configuradas através de interfaces do sistema, com a finalidade de atender a um grande número de indivíduos, respeitando seus limites de velocidade e capacidade de controle do sistema.

Essa é a seqüência básica de todo o sistema: primeiramente, configura-se o ambiente de uso com todas as informações possíveis sobre o ele. Depois, realiza-se a execução propriamente dita do software. O sistema fica em background, repetindo todos os procedimentos anteriormente descritos, desde a aquisição da imagem até a localização, reconhecimento e movimento do objeto marcador na prática, todos esses passos são processados em tempo real.

4. Estrutura do Sistema

A modelagem do sistema foi baseada na orientação à objetos descrita em *Rumbaugh, (1994) e Larman (2000)*, que trata os objetos de uma maneira abstrata, utilizando modelos fundamentados em conceitos do mundo real. O modelo utilizado para a modelagem foi o padrão UML.

Inicialmente, temos dois casos de uso do sistema:

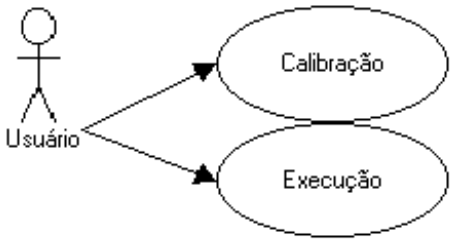


Fig. 4-1 - Diagrama de Caso

Para o caso da calibração temos o seguinte diagrama de classes:

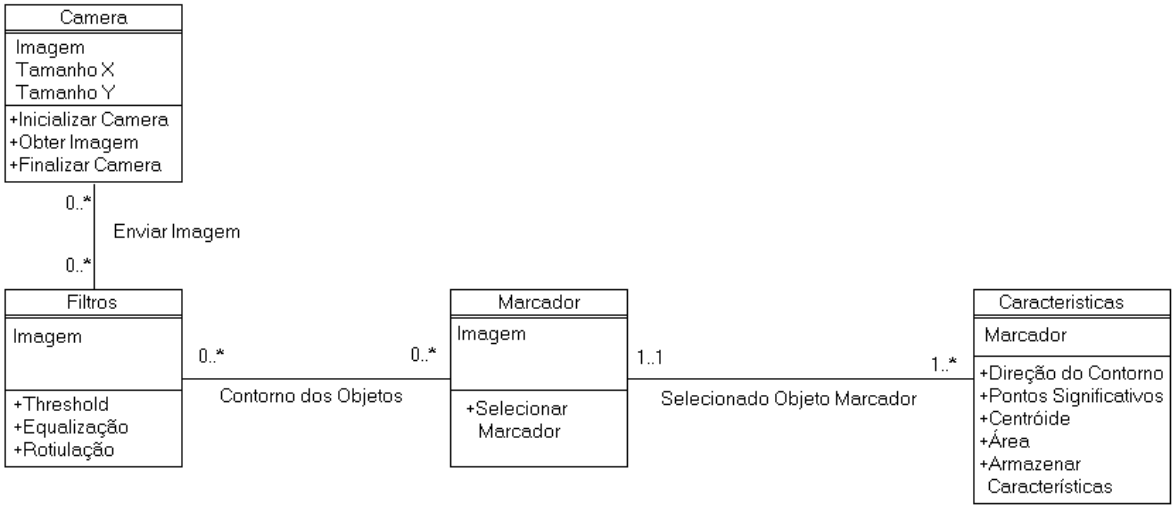


Fig. 4-2 - Diagrama de Calibração

Para o caso da execução do sistema temos o seguinte diagrama de classes:

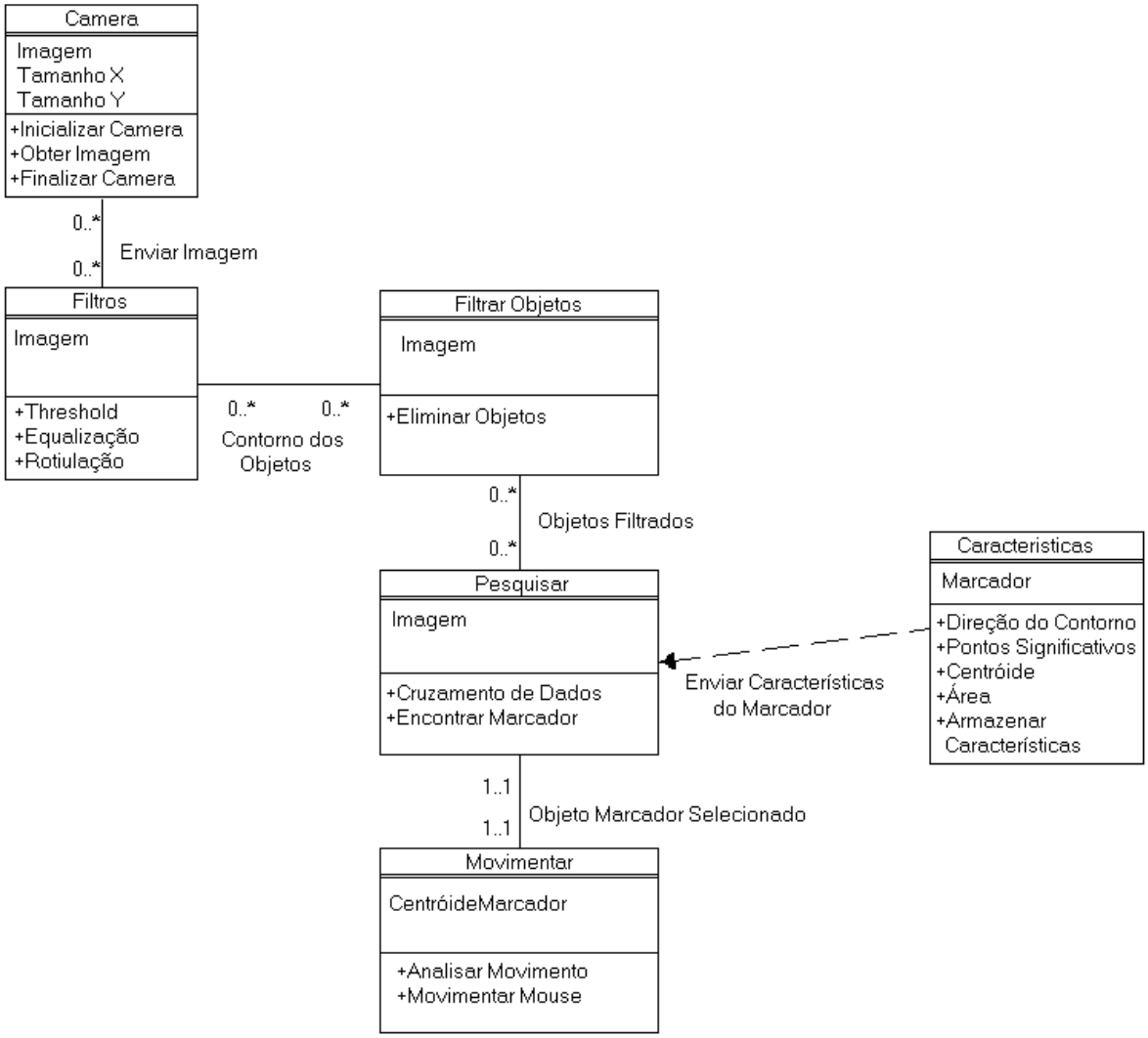


Fig. 4-3 - Diagrama de Execução

A execução deste sistema pode ser representada pelo diagrama de estados, que mostra o fluxo da informação. O diagrama seguinte ilustra a calibração.

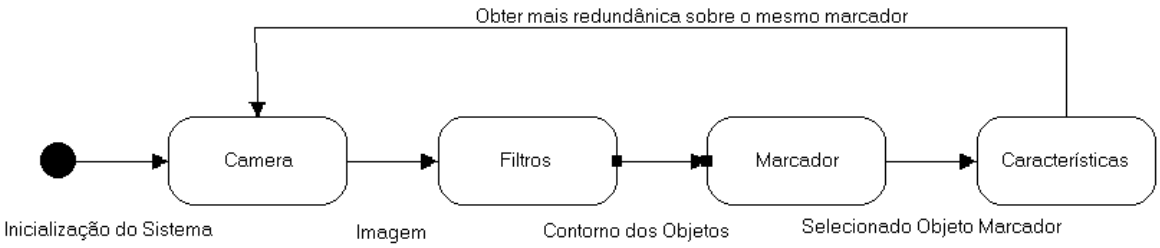


Fig. 4-4 - Diagrama de Estados da Calibração

Outro estado do REMMC é a execução.

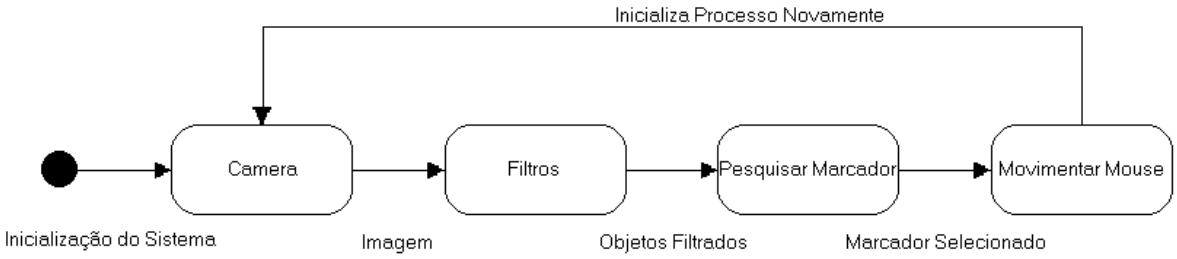


Fig. 4-5 - Diagrama de Estados da Execução

Estes diagramas mostram uma visão geral dos procedimentos executados em cada fase do sistema (Calibração e Execução). Estas classes foram agrupadas em três bibliotecas no sistema Windows®, na forma de DLLs (Dynamic Link Librares): CapCamera (Captura Imagem da Câmera), Filtros (Tratamento da Imagem) e Mouse (Analisar e Movimentar o Mouse). O programa REMMC é quem agrupa e utiliza os recursos disponíveis em cada biblioteca. A seguir, descrevemos os principais métodos de cada biblioteca:

4.1. CapCamera

Para que se obtenha o sinal da câmera, utilizam-se algumas funções especiais do Windows®, as APIs – Application Programming Interface ou Interface de Programação de Aplicativos – que são rotinas ou funções pré-compiladas e prontas (normalmente na forma de dll) para realizarem determinadas tarefas no sistema. Estas interfaces foram concebidas para padronizar os recursos do sistema operacional utilizados por outros aplicativos

O sistema operacional Windows® utiliza troca de mensagens entra todas as aplicações e com o próprio sistema operacional, mas as aplicações que quiserem trocar algum tipo de mensagem com o Windows® devem usar uma função especial, chamada de SendMessage, que é responsável por enviar mensagens ao Windows® (*Petzold & Yao, 1996; Petzold , 1998*) . Essas mensagens podem ser tanto uma consulta de espaço livre em disco como pode ser uma mensagem de ativação de um dispositivo externo, por exemplo, uma câmera de vídeo. Em nossa aplicação, o sistema se comunica através de mensagens com a DLL de vídeo ("avicap32.dll"). Deve ser observado que, se estamos utilizando

drivers do próprio sistema operacional, isso quer dizer que se a câmera funcionar no Windows®, funcionará também em nossa aplicação

O método construtor da classe CapCamera inicialmente obtém informações do driver, versão e tamanho da imagem, utilizando o método "CapGetDriverDescriptionA". Em seguida, faz-se a ligação do driver da câmera com algum objeto da aplicação através do método "CapCreateCaptureWindowA", ambas funções da DLL "avicap32" do sistema operacional Windows®.

Depois da conexão do driver da câmera com algum objeto da aplicação, são realizadas algumas verificações dos recursos e capacidades da câmera. E ,por fim, é informado ao driver da câmera (através de mensagens do Windows®) a função que será chamada a cada nova imagem montada na memória. Isso quer dizer que, quando a imagem estiver pronta na memória, a função receberá do Windows® um sinal informando que a imagem está disponível, neste caso será o método *MyFrameCallBack,* que se encontra dentro da classe CapCamera no sistema REMMC.

Além do controle de mensagens do Windows®, a biblioteca também está relacionada com todo o controle da câmera de vídeo, como por exemplo brilho e contraste da Imagem, velocidade de captura da imagem, quadros por segundo, etc. Esta classe foi implementada utilizando a linguagem Visual Basic® da Microsoft®.

Os métodos são:

4.1.1. Controle da Câmera de Vídeo

- Formato (Configura o tamanho e o formato de aquisição do vídeo)
- Source (Configuração da câmera para ajustes diversos, por exemplo brilho e contraste)
- Compression (Modos de compressão da imagem)

4.1.2. Captura da Imagem

- InicializarCamera (Inicializar a captura do sinal de vídeo e colocá-lo em um formulário)
- FinalizarCamera (Finaliza o sinal de vídeo, limpando da memória todas as informações referentes ao sistema)

- GetSizeX, GetSizeY (Obter o tamanho da imagem)
- SetSizeX, SetSizeY (Setar um novo tamanho para a imagem)
- ExisteERRO (Inicializa uma variável interna para armazenar possíveis problemas com a captura do sinal de vídeo)
- BytesUsados (Retorna o tamanho da imagem em bytes)

4.1.3. Tratamento da lista de objetos da Imagem

- CutPattern (Faz a comparação entre duas cadeias, neste caso contornos)
- PaintGray (Imprimir a matriz em um formulário, mas em níveis de cinza)
- PaintColor (Imprimir a matriz em um formulário, mas colorida)
- PaintLabelColor (Imprimir a matriz rotulada utilizando cores fortes para diferenciar os objetos)
- PaintValueMatrixGray (Imprimir a matriz na forma de texto)

4.2. Filtros

Esta biblioteca concentra os algoritmos de processamento de imagem. A grande maioria das funções recebe um ponteiro de memória para o endereço da matriz da imagem adquirida a partir da câmera de vídeo. Com este endereço, as funções podem aplicar os filtros diretamente na imagem, reduzindo o consumo de memória e tempo de acesso à informação. Podemos dizer que essas funções representam o núcleo do sistema, porque todos esses procedimentos devem ser executados praticamente em tempo real. Dentro desta necessidade, a classe foi construída utilizando a linguagem Visual C++® .

Segue o detalhamento dos algoritmos mais importantes.

4.2.1. Equalização de Histograma

É uma maneira de manipulação de histograma que reduz automaticamente o contraste em áreas muito claras ou muito escuras de uma imagem. Expande também os níveis de cinza ao longo de todo o intervalo. Consiste em uma transformação não-linear, que considera a distribuição acumulativa da imagem original, para gerar uma imagem resultante, cujo histograma será aproximadamente uniforme, conforme Fig. 4-6.

A opção de equalização parte do princípio de que o contraste de uma imagem seria otimizado se todos os 256 possíveis níveis de intensidade fossem igualmente utilizados ou,

em outras palavras, todas as barras verticais que compõem o histograma fossem da mesma altura. Obviamente isso não é possível devido à natureza discreta dos dados digitais de uma imagem de sensoriamenro. Contudo, uma aproximação é conseguida ao se espalhar os picos do histograma da imagem, deixando intocadas as partes mais "chatas" do mesmo. (*Hummel, 1975; Gonzales & Fittes 1977 ; Woods & Gonzales, 1981*).

A equalização de histograma preserva a ordem dos pixels do preto para o branco na escala de cinza e o resultado da função é uma melhor distribuição dos pixels, conforme mostrado na Fig. 4-6.

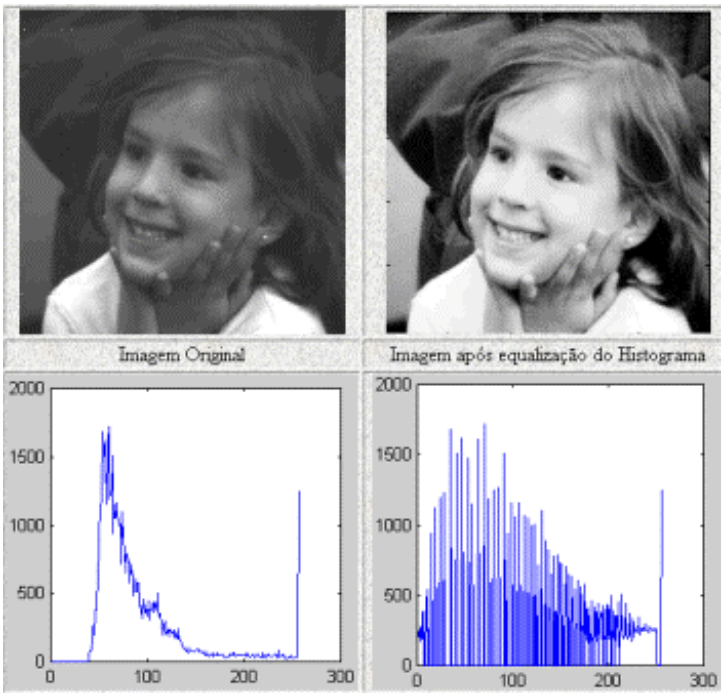


Fig. 4-6 - Equalização do Histograma

4.2.2. Threshold

Existem várias maneiras de se realizar o fatiamento dos níveis, mas a maioria delas são variações de duas básicas.

Uma delas consiste em clarear uma faixa desejada de níveis de cinza preservando o fundo e tonalidades de níveis de cinza na imagem.

A outra consiste em substituir um alto valor *k* para todos os níveis de cinza dentro de uma faixa de interesse e um baixo valor para todos os outros níveis de cinza. Essa transformação resulta numa imagem binária (mostrada na Fig. 4-7). Essa função é responsável por transformar a imagem de entrada em uma imagem binária. (*Gonzales & Woods, 1992*)

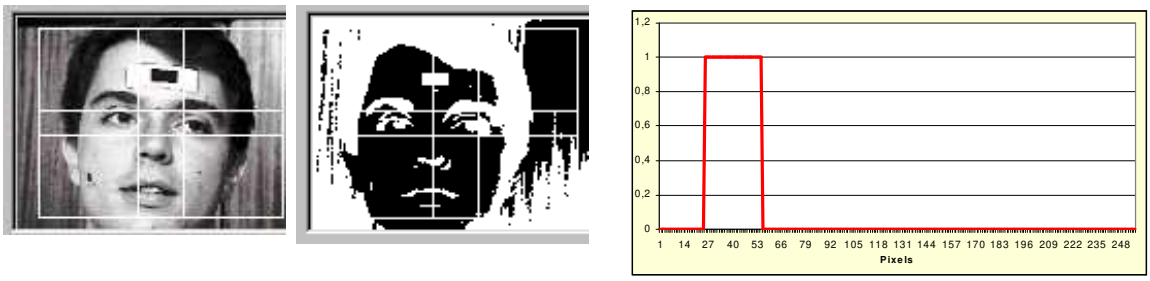


Imagem Níveis de Cinza | Imagem Binária | Corte do Histograma Nível=40 , Janela=30

Fig. 4-7 – Binarização da Imagem usando Threshold

Neste exemplo a faixa de valores no intervalo de 25-55 é representado com o valor máximo, e o restante dos pixels com o valor mais baixo. O exemplo ilustra a posição inicial marcada como 40 e uma janela de abertura deste ponto para mais e para menos de 15 unidades, metade do valor da janela passada a função.

4.2.3. Convolução (Filtros usando Máscara)

Convolução é uma das operações mais comuns no domínio espacial realizada sobre imagens. Um núcleo (*kernel*) de números é multiplicado para cada pixel e seus vizinhos em uma pequena região, os resultados são somados e a soma é armazenada na posição equivalente da imagem resultante. Isto é aplicado para todos os pixels da imagem.

Este tipo de convolução é particularmente comum para suavização (*smoothing*) e operações derivadas. Um exemplo de núcleo seria:

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

Esta operação de convolução com um núcleo pequeno (por exemplo 3x3) é rápida. Aplicar um núcleo maior torna a operação mais lenta. A partir de um certo tamanho de núcleo compensa transformar a imagem para o domínio da freqüência via transformada de Fourier. O tempo necessário para realizar uma transformação de Fourier (FFT) do domínio espacial para o domínio da freqüência mais o tempo da convolução no domínio da freqüência passa a ser menor que a convolução somente no domínio espacial.

Seja a imagem representada por f(x, y) e o núcleo representado por g(x, y). As transformadas de Fourier de f(x, y) e g(x, y) são, respectivamente, F(u, v) e G(u, v). O símbolo * representa a operação de convolução. Com base nessa notação tem-se a seguinte equivalência (*Gonzales & Woods, 1992*):

$$f(x, y) * g(x, y) \Leftrightarrow G(u, v)\, F(u, v)$$

A convolução no domínio da freqüência resume-se a uma multiplicação de matrizes, que é feita da seguinte forma:

1. Transforma-se a imagem para o domínio da freqüência.
2. Transforma-se o núcleo para o domínio da freqüência.
3. Efetua-se a multiplicação de matrizes.
4. Transforma-se a imagem para o domínio do espaço.

Deve-se observar que na convolução no domínio espacial não realizamos a operação nos pixels próximos da borda da imagem, pois não há pixels vizinhos. Às vezes criamos um núcleo especial para estes pixels. No domínio da freqüência, não é necessário qualquer arranjo. Note que a equivalência entre multiplicação no domínio da freqüência e convolução no domínio espacial é restrita a filtros lineares ou multiplicativos.

Na figura abaixo, aplicamos um outro formato de máscara chamado de cômputo Laplaciano (*Marr & Hildreth, 1980*). Neste exemplo, a imagem resultado mostra o contorno do objeto, isto porque a imagem original é binária.

Imagem Original **Resultado** **Máscara (3x3)**

```
 0   -1    0
-1    4   -1
 0   -1    0
```

Fig. 4-8 - Aplicação do Filtro Cômputo Laplaciano

4.2.4. Rotulação

A rotulação de imagem se baseia na conectividade entre pixels, que é um conceito importante usado no estabelecimento de bordas de objetos e componentes de região de uma imagem. Para estabelecer se dois pixels estão conectados , é preciso determinar se eles são de alguma forma adjacentes (vizinhos). A função deste sistema simula o processo de rotulação descrita por *Gonzales & Woods, (1992).*

4.2.4.1. Os vizinhos de um pixel

Como já mencionado, uma imagem será representada por *f(x,y)* . Ao referenciarmos um pixel particular, usaremos letras minúsculas, tais como *p* e *q* . Um subconjunto de pixels de *f(x,y)* será denotado por S.

Um pixel *p* nas coordenadas *(x,y)* possui quatro vizinhos *horizontais e verticais*, cujas coordenadas são dadas por:

$$(x+1,y), (x-1,y), (x,y+1), (x,y-1)$$

Esse conjunto de pixels, chamado *vizinhança-de-4* de *p*, é representado por $N_4(p)$. Cada pixel está a uma unidade de distância de *(x,y)*, sendo que alguns vizinhos de *p* ficarão fora da imagem digital se *(x,y)* estiver na borda da imagem.

Os quatro vizinhos diagonais de p possuem como coordenadas

(x+1,y+1), (x+1,y-1), (x-1,y+1), (x-1,y-1)

E são denotados por $N_D(p)$. Esses pontos, juntos com a *vizinhança-de-4*, são chamados de *vizinhança-de-8* de *p*, representada por $N_8(p)$. Como antes, alguns dos pontos de $N_D(p)$ e $N_8(p)$ cairão fora da imagem quando *(x,y)* se encontrar na borda da imagem.

4.2.4.2. Conectividade

A conectividade entre pixels é um conceito importante usado no estabelecimento das bordas de objetos e componentes de regiões em uma imagem. Para estabelecer se dois pixels estão conectados, é preciso determinar se eles são de alguma forma adjacentes (digamos, se são *vizinhos-de-4*) e se seus níveis de cinza satisfazem um certo critério de similaridade (digamos, se eles são iguais). Por exemplo, em uma imagem binária de valores 0 e 1, dois pixels podem ser *vizinhos-de-4*, mas eles só são ditos conectados, se tiverem o mesmo valor.

Seja *V* um conjunto dos valores de níveis de cinza usados para definir conectividade; por exemplo, em uma imagem binária, $V = \{1\}$ para a conectividade de pixels com valor 1. Em uma imagem em níveis de cinza, para conectividade de pixels com uma escala de valores de intensidade, digamos entre 32 e 64, segue que $V=\{32,33,...,63,64\}$. Consideramos dois tipos de conectividade:

(a) *conectividade-de-4*. Dois pixels *p* e *q*, assumindo valores em *V*, são conectados-de-4 se *q* está no conjunto $N_4(p)$.

(b) *conectividade-de-8*. Dois pixels *p* e *q*, assumindo valores em *V*, são conectados-de-8 se *q* está no conjunto $N_8(p)$.

4.2.4.3. Rotulação de componentes conexos

Vamos considerar uma imagem que seja percorrida pixel por pixel, da esquerda para a direita e de cima para baixo, e assumir, por enquanto, que estamos interessados em componentes *conectados-de-4*. Seja *p* o pixel em qualquer passo no processo de varredura e sejam *r* e *t* , respectivamente, os vizinhos superior e esquerdo de *p*. A natureza da seqüência de varredura garante que, quando chegarmos a *p*, os pontos *r* e *t* já tenham sido encontrados (e rotulados se tiverem valor igual a 1).

Tendo estabelecido os conceitos, vamos considerar o seguinte procedimento. Se o valor de *p* é 0, mova para a próxima posição. Se o valor de *p* é 1, examine *r* e *t*. Se ambos forem 0, atribua a *p* um novo rótulo (com base na informação corrente, esta é a primeira vez que o componente conexo foi encontrado). Se apenas um dos dois vizinhos for 1, atribua a *p* o seu rótulo. Se ambos forem 1 e possuem o mesmo rótulo, atribua a *p* aquele rótulo. Se ambos forem 1, mas possuem rótulos diferentes, atribua um dos rótulos a *p* e anote que os dois rótulos são equivalentes (isto é, os pontos *r* e *t* estão conectados por *p*). Ao fim da varredura todos os pontos com valor 1 terão sido rotulados, mas alguns destes rótulos poderão ser equivalentes.

Agora só nos resta ordenar todos os pares de rótulos equivalentes em classes de equivalência, atribuir um rótulo diferente a cada classe e então percorrer a imagem novamente, trocando-se cada rótulo pelo rótulo atribuído a classe de equivalência.

A rotulação de componentes *conectados-de-8* se faz da mesma maneira, mas os dois vizinhos diagonais superiores de *p*, denotados de *q* e *s*, devem também ser examinados (*Gonzales & Woods, 1992*).

No exemplo abaixo, temos uma imagem binária com três objetos, note que a imagem resultado coloca um número diferente para cada objeto conexo.

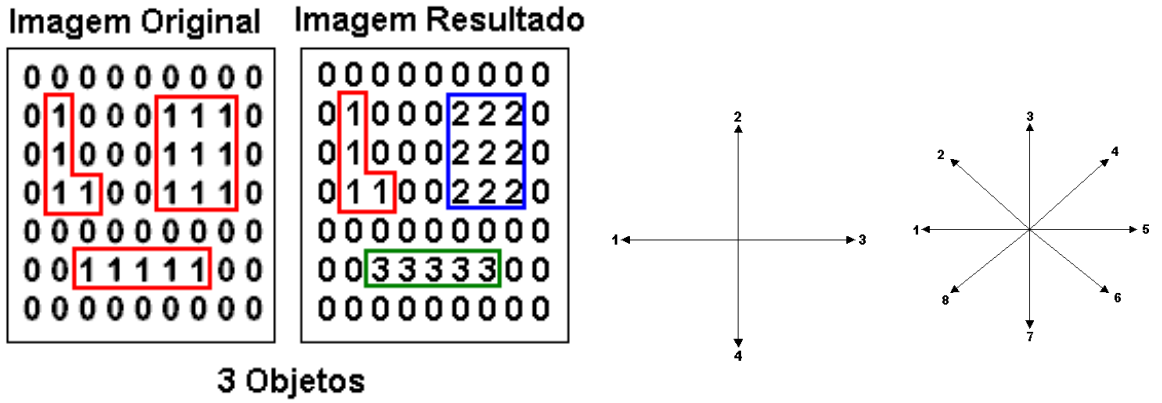


Fig. 4-9 - Rotulação de Imagem

Neste outro exemplo, usamos vários marcadores para observarmos a rotulação de cada objeto conexo e cada um tem uma cor diferente para facilitar a visualização dos objetos, mas na execução do sistema cada objeto tem um índice incrementado de um em um.

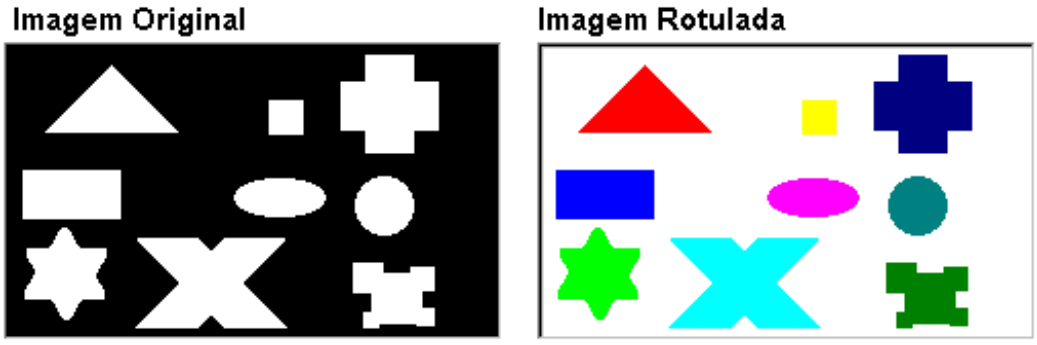


Fig. 4-10 - Rotulação de Vários Objetos

4.2.5. FindPixel

Segmentar consiste na realidade em dividir a imagem original em diferentes regiões, que serão posteriormente analisadas por algoritmos especializados em busca de informações ditas de "alto-nível".

Essa função FindPixel retorna uma matriz preenchida apenas com os pixels iguais aos que são passados como argumentos para ela. Este procedimento é muito utilizado pelo sistema para a realização do processo de segmentação, isto é, isolar objeto por objeto.

Conhecendo a quantidade de objetos rotulados, basta percorrer todo o intervalo do número de objetos e, a cada iteração, selecionar um objeto diferente. A utilização desta função é parecida com a função Find utilizada no MatLab *(LITTLEFIELD & HANSELMAN, 1999)*.

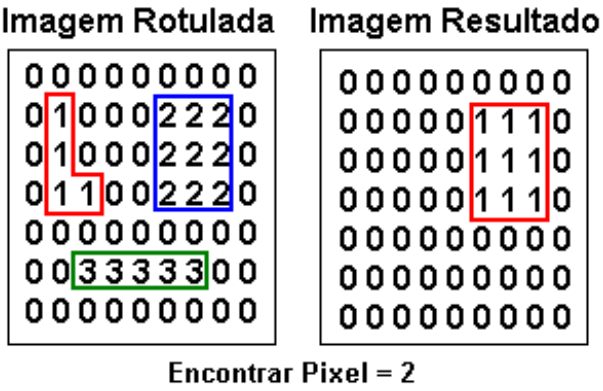


Fig. 4-11 - Busca de Objetos dentro da Imagem

4.2.6. Código de Cadeia

A representação pelo código de cadeia foi primeiramente proposto por *Freeman (1961, 1974)*. Dado que uma imagem tenha sido segmentada em regiões (objetos conexos), usando técnicas para dividir os objetos rotulados, podemos representar essas regiões em termos de suas características externas (sua fronteira).

A representação do contorno de um objeto consiste na retirada dos pixels que compõem a sua fronteira (a borda do objeto), sendo que esses pixels podem estar conectados por uma *conectividade-de-4* ou *conectividade-de-8* . A direção de cada segmento é codificada por um esquema de numeração como aquele mostrado na Fig. 4-12.

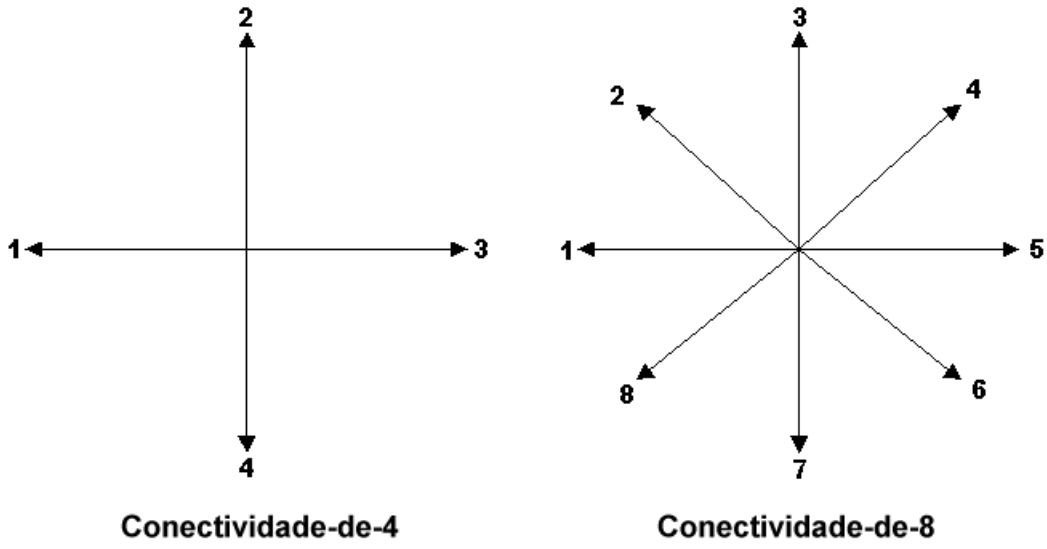


Fig. 4-12 - Direções do Contorno

As imagens digitais são usualmente adquiridas e processadas no formato de uma grade com espaçamento igual nas direções *x* e *y*, de maneira que um código da cadeia possa ser gerado seguindo-se a fronteira do objeto, por exemplo, no sentido horário e atribuindo-se uma direção aos segmentos que conectam cada par de pixels.

No exemplo abaixo, retiramos o contorno de cada objeto utilizando *conectividade-de-4* e *conectividade-de-8*. O resultado deste algoritmo é simplesmente o contorno do objeto.

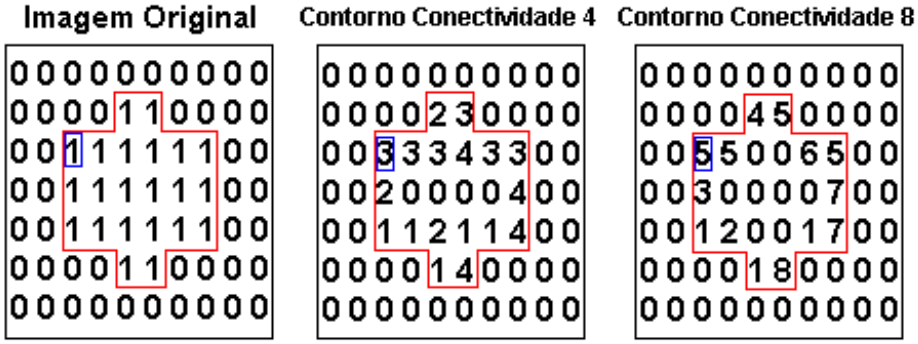


Fig. 4-13 - Retirando o Contorno de uma Imagem

No exemplo da Fig. 4-13, temos uma imagem com um único objeto na forma de cruz, o seu ponto inicial está marcado em azul. No primeiro contorno temos a forma do objeto usando a *conectividade-de-4* e no segundo exemplo temos a *conectividade-de-8*.

Neste outro exemplo ilustramos o contorno de um objeto e a representação do contorno segue o sentido da *conectividade-de-8*.

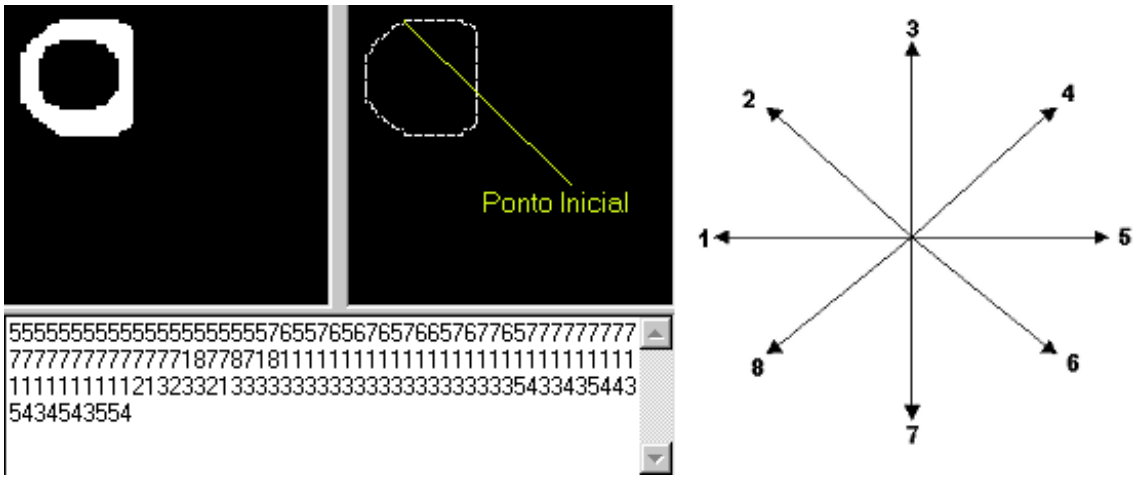


Fig. 4-14 - Exemplificando a Cadeia de Caracteres de um Contorno

4.2.7. Identificar a Direção do Contorno (VarDir)

Dado o contorno de um objeto pela função Código de Cadeia - 4.2.6, reescrevemos a cadeia de caracteres que representa a forma do objeto em uma outra cadeia de caracteres, contendo somente as variações de direções do contorno no sentido horário, anti-horário e quando não existe a mudança de direção (por exemplo uma reta).

Seguindo as direções definidas, (Fig. 4-12 - Direções do Contorno), percorremos o contorno da esquerda para a direita, comparamos um caractere do contorno com o próximo e analisamos se houve ou não mudança de direção.

A identificação das direções do contorno são representadas por três caracteres diferentes (*Serapião, 1997*):

- Z (não existe mudança do sinal)
- P (mudança no sentido anti-horário)
- N (mudança no sentido horário)

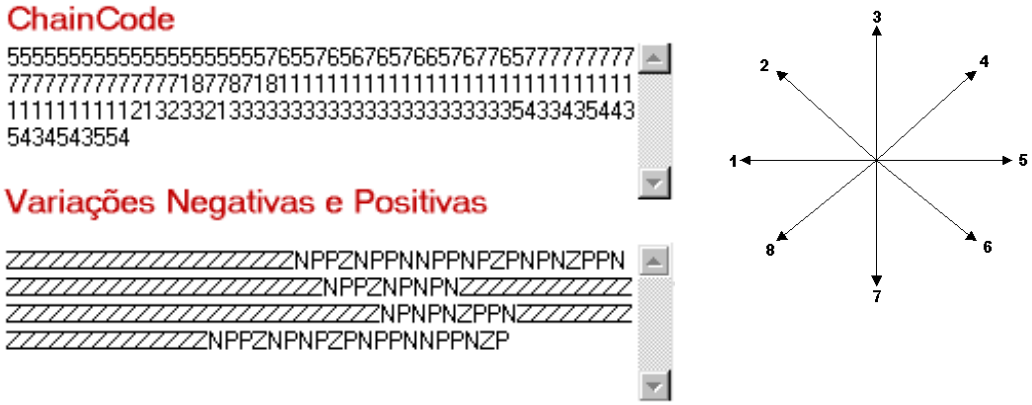


Fig. 4-15 - Identificação de Pontos de Mudança no Contorno

4.2.8. Localizar Pontos Significativos (Psig)

Tendo o contorno do objeto mais as variações de direções desse contorno, basta comparar as duas cadeias e filtrar somente os pontos do contorno que representarão os pontos significativos do objeto.

Esses pontos significativos podem ser encontrados percorrendo a cadeia que representa as variações de direção e quando existir os valores de *N* ou *P* basta pegar a mesma posição na cadeia de contorno. A única exceção é o ponto inicial.

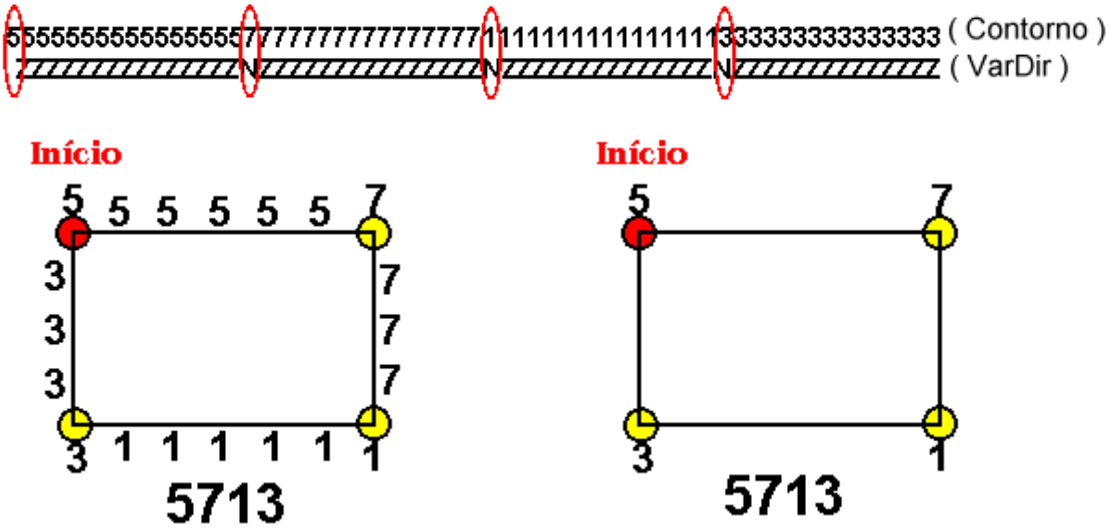


Fig. 4-16 - Pontos Significativos de um Contorno

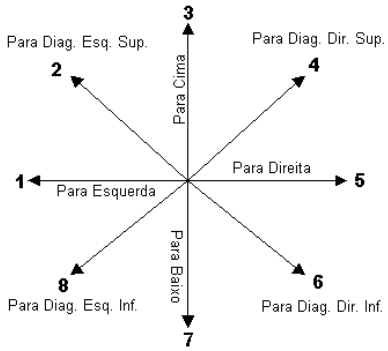
4.3. Mouse

Da mesma forma que na captura da imagem, pode-se também interceptar o controle do mouse. Neste caso, informamos ao sistema operacional a nova coordenada do ponteiro do mouse, através de instruções que são colocadas na fila de mensagens do Windows®.

Quando o sistema operacional processar a mensagem, o cursor do mouse se movimentará para a posição desejada. Esta classe foi desenvolvida utilizando o Visual Basic®.

4.3.1. MoveMouse

Movimenta o mouse na direção desejada. Essa direção deve seguir a tabela de seqüência igual à conectividade 8. Por exemplo, se fornecermos o valor 1 para a função, então o ponteiro do mouse irá deslocar-se para a esquerda.



2	3	4
1	0	5
8	7	6

Movimentos do Mouse Quadro de Reconhecimento

Fig. 4-17 - Quadro de Movimentos do Mouse

Como mostra a figura acima, basta informar a direção de movimento para que a função se encarregue de movimentar o mouse para a direção desejada.

Quando o marcador é selecionado, o sistema confronta a posição encontrada com o quadro de reconhecimento e transmite a localização do marcador para a função MoveMouse, que será responsável por mover o mouse na direção desejada.

4.3.2. EventClick

Esta função simula o pressionamento de um botão do mouse. Internamente, a função envia uma mensagem ao Windows® para ativar o click do mouse. No sistema REMMC, o código 030 representa um click do botão esquerdo. A função recebe um argumento que identifica qual botão será simulado (botão da esquerda, do meio ou da direita).

4.3.3. EventDoubleClick

É um procedimento similar à função anterior, com a única diferença de que esta função simula o duplo click do mouse. No sistema REMMC, o código 070 representa o duplo click do botão esquerdo.

4.3.4. CentralizarMouse

Coloca o ponteiro do mouse no centro do monitor, este método é utilizado no início do programa, a fim de ajustar o ponteiro do mouse sempre no centro do monitor para facilitar o início da execução do sistema.

4.3.5. Velocidade Mouse

Controla o tamanho do salto que o ponteiro do mouse dá de uma posição para outra no vídeo. Quanto maior a velocidade, maior será o salto de uma posição para outra.

Neste capítulo, descrevemos as principais funções usadas no sistema, apresentando uma visão geral dos procedimentos e recursos utilizados no sistema REMMC, incluindo aquisição, filtragem e processamento das informações, geradas a partir de uma imagem capturada por uma câmera de vídeo.

5. Ambiente de Trabalho do Software

O conteúdo deste capítulo descreve as principais janelas e ferramentas para a configuração e operação do sistema, dando uma visão geral de todos os recursos disponíveis em tempo de execução ou calibração.

O ambiente de trabalho do sistema agrupa todas as bibliotecas e ferramentas necessárias para serem utilizadas na captura, tratamento e reconhecimento da imagem. Todas as janelas de configuração estão disponíveis para um melhor ajuste do software.

Todo o controle e a comunicação do software com o sistema operacional é realizado de maneira transparente ao usuário.

5.1. REMMC (Reconhecimento Especial de Marcadores para o Manuseio de Computadores)

Para utilizar todos os recursos, criou-se um ambiente de trabalho, em que se juntam todas as bibliotecas e funções para o adequado funcionamento dos componentes. Este ambiente é o que se chama de REMMC. Neste programa, é possível configurar um marcador, ajustar o ambiente, fazer alguns ensaios sobre a captura da imagem da câmera e todas essas configurações podem ser salvas para posteriormente serem utilizadas, sem a necessidade de reconfigurar todo o sistema.

O REMMC pode ser chamado de "front-end", porque este aplicativo agrupa todas as bibliotecas descritas no capítulo anterior de uma forma ordenada para facilitar a construção, a manutenção e o uso do software no ambiente Windows®. Por este motivo, o sistema foi desenvolvido na linguagem Visual Basic®.

Formulário Principal do REMMC :

Fig. 5-1 - Tela Principal do REMMC

5.1.1. Pré filtragem dos objetos

Neste quadro, conforme a Fig. 5-1, é possível definir uma pré-filtragem dos objetos. Para este exemplo, não serão aceitos os objetos conexos que estiverem abaixo do Mínimo 169 pixels nem acima do Máximo 394 pixels. Com isso o programa descarta de início todos os objetos grandes e pequenos.

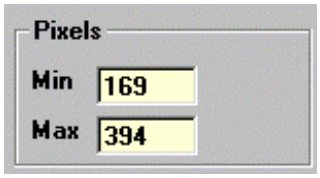


Fig. 5-2 - Configurando o Tamanho do Objeto

Quando o sistema encontra o marcador, ele próprio calcula a área do objeto marcador e fornece ao sistema uma tolerância de 50 % para mais e para menos. Esse procedimento somente é realizado na calibração inicial do marcador. Se o usuário quiser, pode modificar esses dados manualmente.

5.1.2. Inversão da Imagem

Em alguns casos a imagem pode estar invertida principalmente pela configuração da câmera ou pela sua posição. Tendo em vista a versatilidade de uso do sistema, acrescentaram-se duas características ao programa que é a possibilidade de inverter a imagem no eixo X ou Y.

Fig. 5-3 - Configuração dos Eixos da Imagem

5.1.3. Quadro de Reconhecimento

Quando o sistema selecionar o marcador no meio de todos os outros objetos da imagem, automaticamente este objeto marcador será colocado em quadro para reconhecer a sua posição. Em seguida será calculada a diferença entre a posição anterior e a posição atual, podendo assim informar o seu deslocamento.

Com o treinamento e a experiência no uso do sistema, o usuário poderá melhorar a sensibilidade do movimento, reduzindo as medidas do quadro de reconhecimento e aumentando a velocidade do mouse.

Este frame do formulário configura a sensibilidade do quadro de reconhecimento.

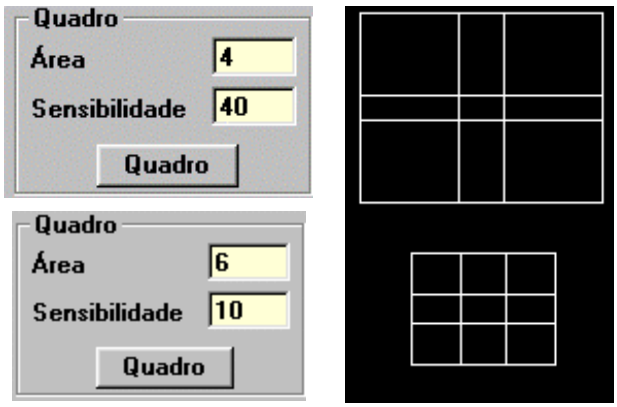


Fig. 5-4 - Configurando a Sensibilidade do Movimento

O campo "área" informa ao sistema o quanto da área total do quadro de reconhecimento será utilizada, a faixa de valores vai de 1 até 10. Movimentos bruscos podem ser configurados com uma maior área de detecção.

O campo "sensibilidade" configura a área central do quadro de reconhecimento, a faixa de valores vai de 1 até 100. Esta opção configura movimentos curtos ou movimentos longos.

5.1.4. Configuração do Mouse

Neste frame, pode-se configurar a velocidade do mouse, isto é, ajustar o tamanho do salto de um ponto para outro do vídeo. A repetição consiste em determinar ao sistema quando ele deve interpretar uma posição do marcador como movimento ou controle. Por exemplo, para movimentar o mouse para a direita, é preciso levar o marcador para a direita e esperar 5 ciclos para que, então, o mouse comece a se movimentar. Para encerrar o movimento basta retornar o marcador para o centro.

Para citar outro exemplo, observe-se o funcionamento do click do mouse. Para simulá-lo basta o usuário levar o marcador do centro para cima e voltar para o centro. Cada movimento do marcador tem que ser, no mínimo, menor que o ciclo da "Repetição" para o sistema interpretar como controle.

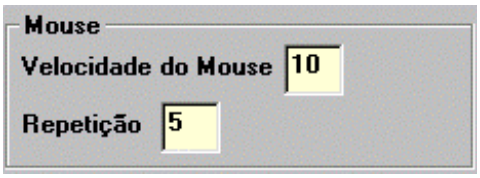


Fig. 5-5 - Configurando a Velocidade do Mouse

O ciclo de repetição define o tamanho das palavras de código que podem ser utilizadas para conhecer eventos que não correspondem à movimentação do cursor

5.1.5. Configuração dos Limites de Binarização da Imagem

Neste item, pode-se configurar o corte do histograma (nível) e a janela usada. Dado o ponto de corte, o programa calcula um intervalo (janela). Quando o marcador é selecionado, automaticamente o seu tom de cinza é colocado como sendo o nível de corte do histograma e o sistema calcula uma janela de 20%, por exemplo:

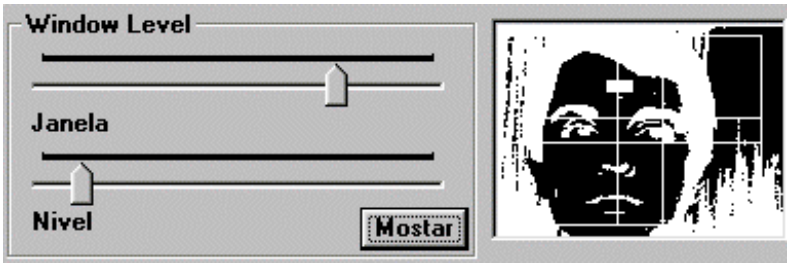


Fig. 5-6 - Configurar a Binarização da Imagem

O *"nível"* nada mais é que um ponto do histograma. A *"janela"* é o intervalo entre "Nível - (Janela/2)" e "Nível + (Janela/2)"; todos os pixels nesta faixa ficam brancos *(255)* e o restante dos pixels ficam preenchidos com preto *(0)*. Este recurso utiliza a função de threshold como foi descrita no item 4.2.2.

5.1.6. Botões

Fig. 5-7 - Botões de Operação

- Capturar: Captura a imagem da câmera transformando-a em níveis de cinza
- Rodar: Inicia a execução do programa conforme as configurações da janela do REMMC, mas a tela principal continua sendo mostrada.
- Aplicar: Mostra um esboço da imagem final, passando todos os filtros e acertos configurados no sistema.

5.2. Verificação de Marcador

Este utilitário serve para testar todos os atributos de cada marcador. O utilitário é bastante simples e sua principal função é testar, avaliar e validar a escolha do melhor marcador para um determinado ambiente. Exemplo:

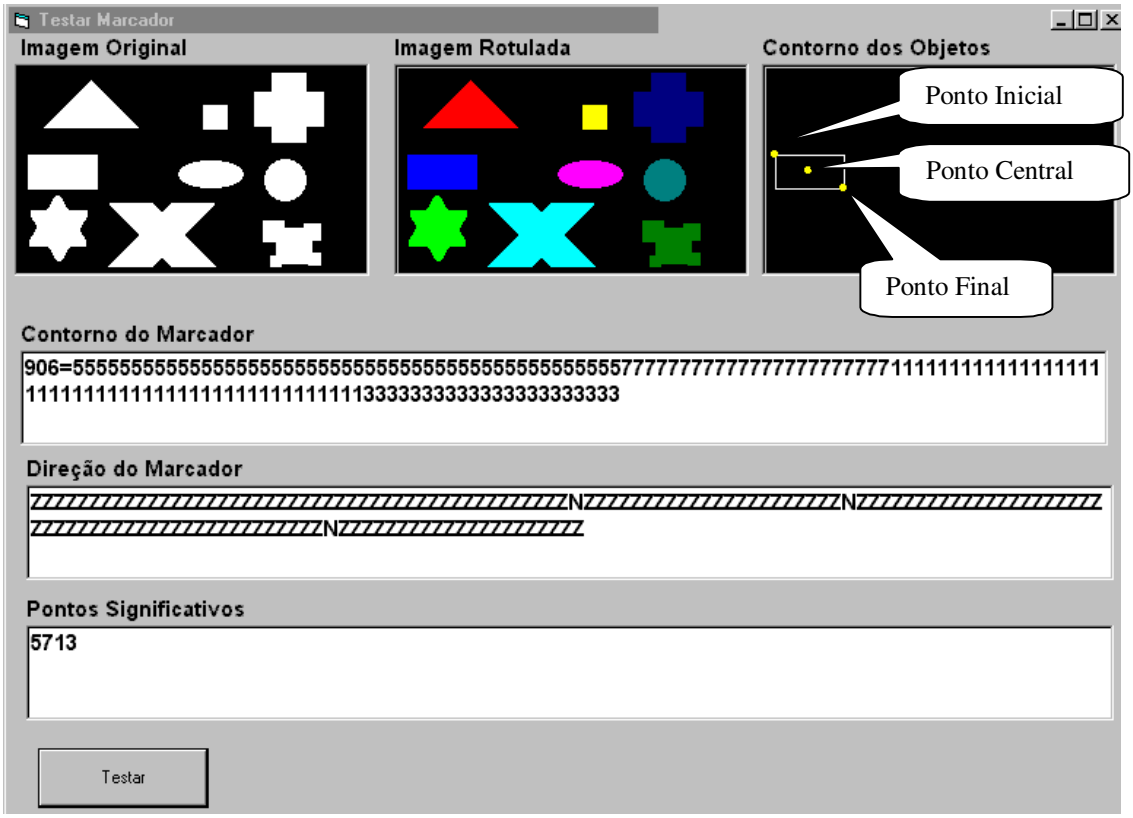


Fig. 5-8 - Identificação do Marcador

Na figura abaixo, o contorno do objeto *cruz* tem muito mais características do que o objeto da Fig. 5-8. Comprova-se este fato, comparando-se os pontos significativos deste objeto com os do anterior.

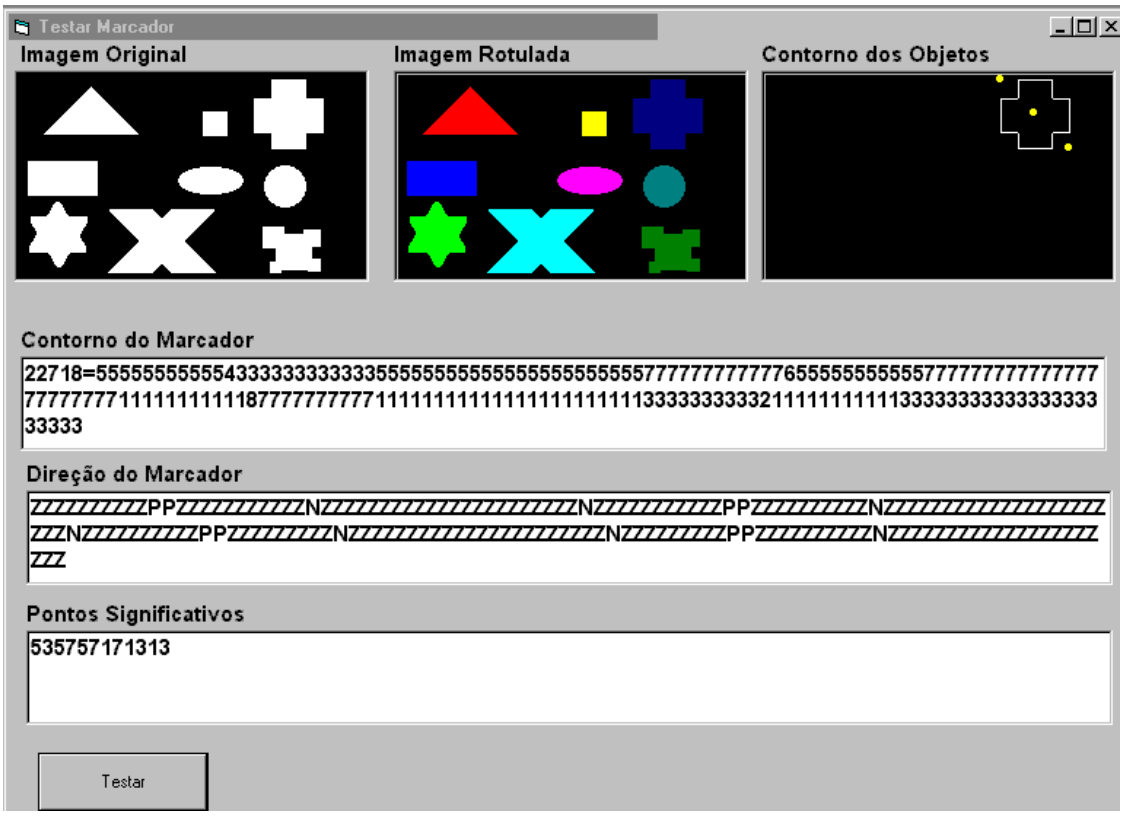


Fig. 5-9 - Outro Exemplo de Marcador

O ambiente de trabalho REMMC disponibiliza ao usuário todo o controle necessário para a configuração e calibração do software. Além de todos os recursos aqui demostrados, o usuário também tem como salvar ou ler uma calibração previamente configurada, facilitando as demais utilizações do sistema.

A grande vantagem é a possibilidade da escolha do melhor marcador, que pode ser auxiliada pelo utilitário anteriormente descrito, ou a simples troca de um marcador por outro, que é um procedimento bastante simples no sistema. Basta calibrar o software com o novo marcador.

6. Reconhecendo o Movimento do Marcador

Neste capítulo, resume-se o procedimento aplicado no sistema para reconhecer um marcador dentre todos os outros objetos contidos na imagem. A pesquisa é realizada de uma forma simples e eficaz, com a finalidade de reduzir ao máximo o processamento computacional para que se atinja o objetivo de localizar o marcador na imagem o mais rápido possível.

6.1. Reconhecimento do Marcador

Reconhecer um marcador em uma outra imagem não é uma tarefa muito fácil. Para realizar esta pesquisa é importante ter algumas características do objeto marcador. Essas informações são adquiridas na calibração do software, na fase inicial. Na Fig. 6-1 ilustrou-se a filtragem da imagem, rotulando os objetos conexos. A partir da imagem rotulada é possível selecionar o objeto marcador e extrair informações para compor a tabela de características (Tabela 6-1).

O reconhecimento do marcador é manual, isto é, o usuário tem que indicar ao sistema qual é o objeto marcador dentro da imagem, isso se dá na fase de calibração.

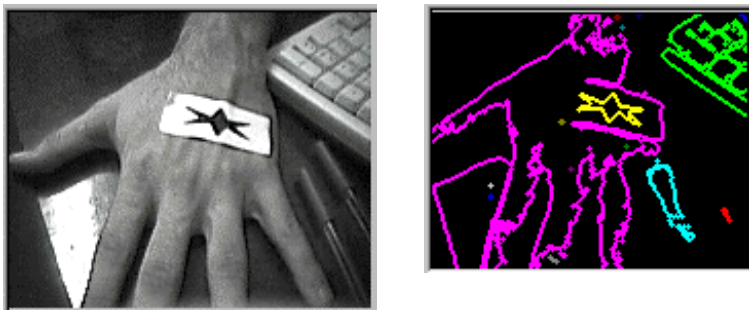


Imagem Original Imagem Rotulada

Fig. 6-1 - Reconhecimento de um Marcador

6.2. Analisando Características da Imagem

Existem duas fases no projeto: a) quando o usuário configura o sistema, passando informações do ambiente, velocidade do mouse, tamanho máximo e mínimo do marcador e informa qual marcador servirá como modelo de pesquisa; b) funcionamento propriamente dito do sistema REMMC. Nesta segunda fase, o sistema recebe da câmera uma imagem, à qual aplica diversos filtros para preencher uma tabela, conforme demonstra o modelo (Tabela 6-1). Esta tabela contém características de todos os objetos da imagem e, para que se encontre o objeto marcador, confrontam-se a tabela de características do marcador, adquirida na fase inicial, com a tabela dinâmica adquirida a partir de cada imagem da câmera. O melhor matching adquirido é o marcador escolhido. Esta técnica foi também aplicada no trabalho descrito por *Serapião* (1997).

Características	**Objeto 1**	**Objeto 2**	**...**	**Objeto X**
Contorno				
PT_Sig				
PT_Sig_Comuns				
Direcao				
Centroide_X				
Centroide_Y				
Diagonal				
PT_Inicial_X				
PT_Inicial_Y				
PT_Final_X				
PT_Final_Y				
TotalPixel				
Area				

Tabela 6-1 - Características dos Objetos

As características da tabela acima podem ser descritas da seguinte forma:

- **Contorno**
  Contorno do objeto utilizando a conectividade 4 ou 8.

- **PT_Sig**
  Pontos significativos do contorno do objeto

- **PT_Sig_Comuns**
  Procurar trechos comuns entre todos os pontos significativos. (Por exemplo: todos os pontos significativos começam com 535).

- **Direção**
  São as mudanças de direções do contorno

- **Centroide_X**
  Ponto central do objeto no eixo X

- **Centroide_Y**
  Ponto central do objeto no eixo Y

- **Diagonal**
  Cálculo da diagonal entre o ponto inicial e o centróide do objeto

- **PT_Inicial_X**
  Ponto inicial no eixo X do objeto em relação ao canto superior esquerdo da imagem

- **PT_Inicial_Y**
  Ponto inicial no eixo Y do objeto em relação ao canto superior esquerdo da imagem

- **PT_Final_X**
  Ponto final no eixo X do objeto em relação ao canto superior esquerdo da imagem

- **PT_Inicial_Y**
  Ponto final no eixo Y do objeto em relação ao canto superior esquerdo da imagem

- **TotalPixel**
  Quantidade de pixels do objeto

- **Área**
  Área total do objeto

6.3. Confrontando as Tabelas

Na fase de calibração do software, são adicionadas à tabela todas as características descritas acima. Da mesma forma, quando o programa está em execução, cria-se uma tabela contendo todas as características de cada objeto, para cada imagem adquirida. Após colher todos os dados, é feita uma comparação entre as duas tabelas para encontrar o marcador. Por exemplo:

- Marcador

Características	**Marcador 1**	**Marcador 2**
Contorno	555777111333	5566111333
PT_Sig	5713	5613
PT_Sig_Comuns		13
Direcao	ZZZNZZNZZNZZ	ZZNZNZZNZZ
Centroide_X	3	3
Centroide_Y	4	4
Diagonal	5	5
PT_Inicial_X	0	0
PT_Inicial_Y	2	2
PT_Final_X	8	7
PT_Final_Y	6	7
TotalPixel	100	110
Area	145	152

Tabela 6-2 - Exemplo de Características de um Marcador

Note que esta tabela tem vários formatos do mesmo marcador porque quanto mais redundância o marcador tiver, maior será o grau de acerto. Isto também acontece no cérebro: *muitas ligações neurais são redundantes, isso acontece para que o processamento cerebral seja mais rápido e preciso*, conforme diz *Rocha* (1999).

- Objetos da Imagem

Características	**Objeto 1**	**Objeto 2**	**Objeto 3**
Contorno	5543355771187711332113 3	...	55676777111333
PT_Sig	5357171313	...	5676713
PT_Sig_Comuns		...	13
Direcao	ZZNNZNZNZNNNZNN ZNNZZ	...	NNZNZNZZZNZZNZZ
Centroide_X	5	...	3
Centroide_Y	7	...	5
Diagonal	8	...	5
PT_Inicial_X	0	...	0
PT_Inicial_Y	5	...	3
PT_Final_X	10	...	8
PT_Final_Y	14	...	7
TotalPixel	183	...	124
Area	207	...	179

Tabela 6-3 - Exemplo de Características da Imagem da Câmera

Com todas as informações previamente obtidas, percorre-se linha por linha da tabela e compara-se o valor do objeto com o valor correspondente do marcador. Consequentemente, seleciona-se o objeto que mais se aproxima das características do marcador. No final, o objeto com mais características parecidas será o escolhido. Neste caso, escolheu-se o objeto 3 da tabela.

6.4. Comparando Cadeias de Caracteres

A comparação de cadeias pode ser feita entre cada elemento da cadeia padrão com cada elemento da cadeia do objeto atual, ou proporcionar um certo salto (*Jump*) de um ponto para outro dentro da própria cadeia. Esse artifício facilita a pesquisa de uma subcadeia dentro da outra, mesmo tendo alguma variação em sua forma original. Neste caso, a função consegue encontrar a cadeia padrão dentro de outra cadeia, mesmo que não sejam exatamente iguais. A opção *Jump* informa a quantidade de elementos que a função deve "olhar" para frente do ponto pesquisado, como mostra a Tabela 6-4

Cadeia Padrão	Cadeia atual	Resposta
613	(6)77(1)2(3) (Jump=1)	Falso
613	(6)77(1)2(3) (Jump=2)	True
613	12(613)9 (Jump=0)	True

Tabela 6-4 - Comparação de Cadeias de Caracteres

Essa técnica é indicada para o caso da cadeia, que descreve a forma do objeto, conter ruídos que possam prejudicar a identificação do marcador.

Uma outra opção ajustável no sistema é a de ângulo. Com esta função, identifica-se o marcador, mesmo que ele esteja rotacionado. Esta mudança de ângulo segue a forma da conectividade 8.

Cadeia Padrão	Cadeia atual	Resposta
613	514 (Ang=0)	Falso
613	514 (Ang=1)	True

Tabela 6-5 - Analisando a Rotação do Objeto

No exemplo acima, obteve-se a cadeia padrão "613" e analisou-se sua forma comparando-a a uma outra cadeia "514". Se as cadeias forem comparadas, o resultado será falso porque não são iguais. Por outro lado, se for feita a análise de uma cadeia, variando uma unidade no seu ângulo com referência à conectividade 8, identifica-se o marcador "613" como sendo igual ao objeto "514" que, provavelmente, estará rotacionado na imagem principal.

6.5. Exclusão de objetos

Existe uma opção para declaração de objetos proibidos, isto é, objetos que não podem ser considerados marcadores. Esta opção pode ser de utilidade em ambientes muito ruidosos, no qual variações de luminosidade ou de movimentações no ambiente possam gerar imagens com certa semelhança com os marcadores selecionados.

Sempre que tais ruídos forem identificados durante o processo de calibração, é possível marcá-los para exclusão.

6.6. Exemplos de Marcadores

Após vários testes, chegamos à conclusão de que o melhor marcador é aquele que tem a cor preta e impresso no centro de uma etiqueta branca. Isso facilita bastante a separação dos objetos. A escolha da cor preta se deve ao fato do nível de cinza do marcador preto estar perto de zero e, mesmo que haja alguma variação na luminosidade da imagem, ele sempre estará em uma região mais escura que as demais.

Outra informação importante sobre o marcador é a quantidade de detalhes, quanto mais detalhes tiver a forma, melhor será o reconhecimento do marcador dentro da imagem. Ainda existe a possibilidade da utilização de outros adesivos infantis ou até mesmo o uso de características da roupa. Veja alguns exemplos utilizados na figura abaixo

Fig. 6-2 - Exemplos de Marcadores

6.7. Reconhecendo o Movimento

A partir do momento em que um marcador é escolhido dentro da imagem, torna-se preciso descobrir qual o movimento e a direção tomada pelo usuário. Para resolver este problema, criou-se um quadro de reconhecimento (descrito anteriormente na seção *5.1.3 – "Quadro de Reconhecimento"*). O processo é bastante simples, basta colocar o ponto central do objeto marcador neste quadro e automaticamente o programa mostrará em qual quadrante se encontra o objeto. Com isso, obtém-se as informações sobre qual posição está o marcador e para onde ele está se movendo. No exemplo abaixo, o objeto está no centro, isto é, a seta do mouse não se movimenta

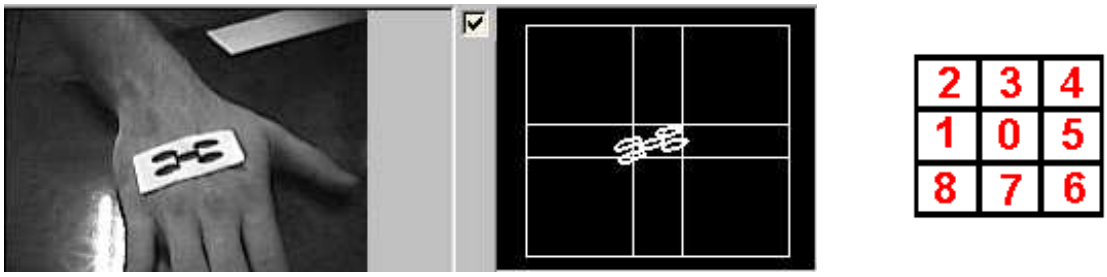


Fig. 6-3 - Reconhecendo o Movimento do Marcador

Abaixo, mostra-se que o marcador foi para a direita. Quando identificado no quadro, descobre-se que a posição atual é a posição 5, que significa "ir para a direita". O marcador é desenhado no quadro para efeito de treinamento e melhor teste do programa, pois se o programa estiver no modo "mínimo visualizador", o objeto marcador será representado apenas pelo seu ponto central. Assim, o sistema fica muito mais rápido.

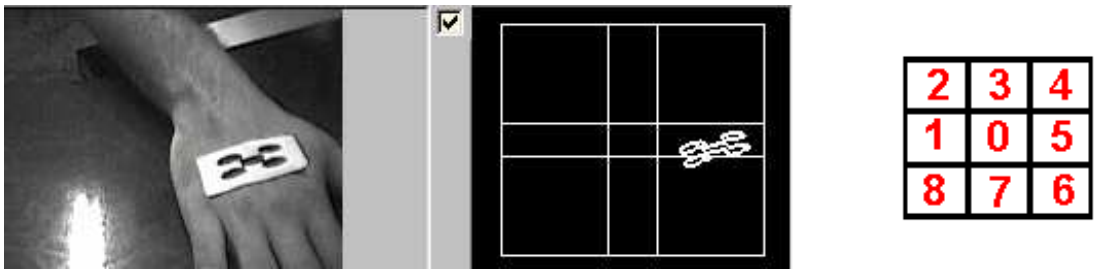


Fig. 6-4 – Movimentando o Mouse para a Direita

6.8. Codificando movimentos

O sistema utiliza os movimentos realizados pelo usuário para criar dois tipos de palavras de códigos com a seguinte estrutura:

1. **(i)^{j} :** onde i ∈ (0,1,2,3,4,5,6,7,8) e **j** <=NR (Número de Repetição) : chamadas *palavras de codificação de controle* (**CC**). e

2. **(i)^{n}** , onde i ∈ (0,1,2,3,4,5,6,7,8) **n**>NR: chamadas *palavras de controle de mouse* (**CM**).

O dígito **O** tem um status especial, pois sinaliza *final da palavra de código.*

Por exemplo: a palavra **1111** ***11111111*** **070** desloca o mouse para a esquerda, a partir da quinta repetição do dígito **1** durante 8 ciclos de processamento (5.1.4 - Configuração do Mouse). Já a palavra 070 codifica um duplo click.

6.9. Utilização Minimizada do REMMC

O REMMC pode, ainda, ser utilizado em background ou em formato minimizado usado para orientar o usuário durante a utilização do sistema operacional Windows®. Para isso, existe um botão chamado de "Mínimo Visualizador", que permite reduzir o tamanho da tela, colocando-o no canto inferior direito do monitor, com uma opção de ficar em cima de qualquer janela aberta no sistema Windows®. Vale lembrar, aqui, que o sistema está rodando em tempo real.

Exemplo da tela minimizada.

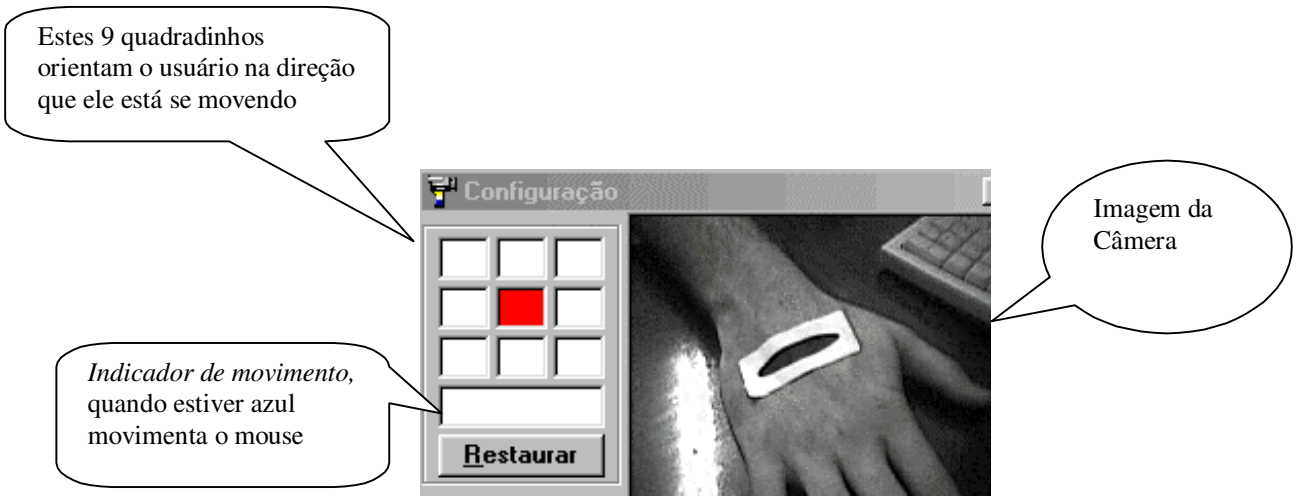


Fig. 6-5 - Utilização Mínima do REMMC

No exemplo da Fig. 6-6, o usuário está com a mão para o lado direito. Sendo assim, o sistema esperará um número de repetições do mesmo movimento, que está pré-definido nas configurações iniciais. Quando o número de repetições desse movimento for maior que o configurado, o sistema mandará um sinal ao sistema operacional para que o mesmo

movimente o mouse na direção deseja. A cadeia desse movimento pode ser representada da seguinte forma: **55555.....55550.**

O início do movimento é sinalizado pela mudança de cor do *indicador de movimento* para a cor azul. Para o usuário encerrar o movimento, basta que retorne o marcador para a posição central, isto é, quando o sistema receber um código de controle **0** (Posição Central), o sistema zera o contador de repetição e pára o movimento do mouse.

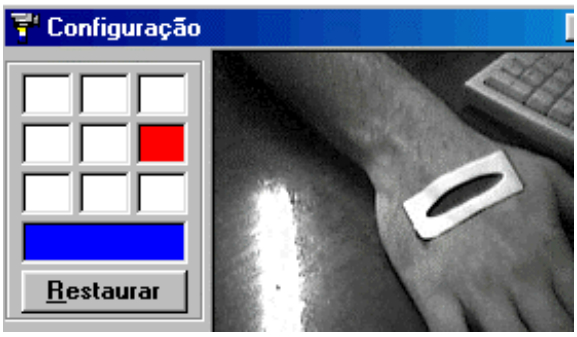


Fig. 6-6 - Utilizando o REMMC na Forma Mínima

Enquanto o *indicador de movimentos* não estiver na cor azul, o software poderá interpretar a cadeia de caracteres como sendo um sinal de controle, por exemplo, um click do mouse. Nesta situação, o usuário poderá emitir sinais previamente codificados para o sistema operacional, a fim de acionar o controle desejado: click, duplo click, etc. Para gerar esse tipo de palavra, o usuário deve retornar o marcador para a posição central antes da mudança de cor do indicador de movimento.

Exemplos de palavras de controles:

Click: Centro, para cima e centro novamente (030)

Duplo Click: Centro, para baixo e centro novamente (070)

Exemplos de cadeias

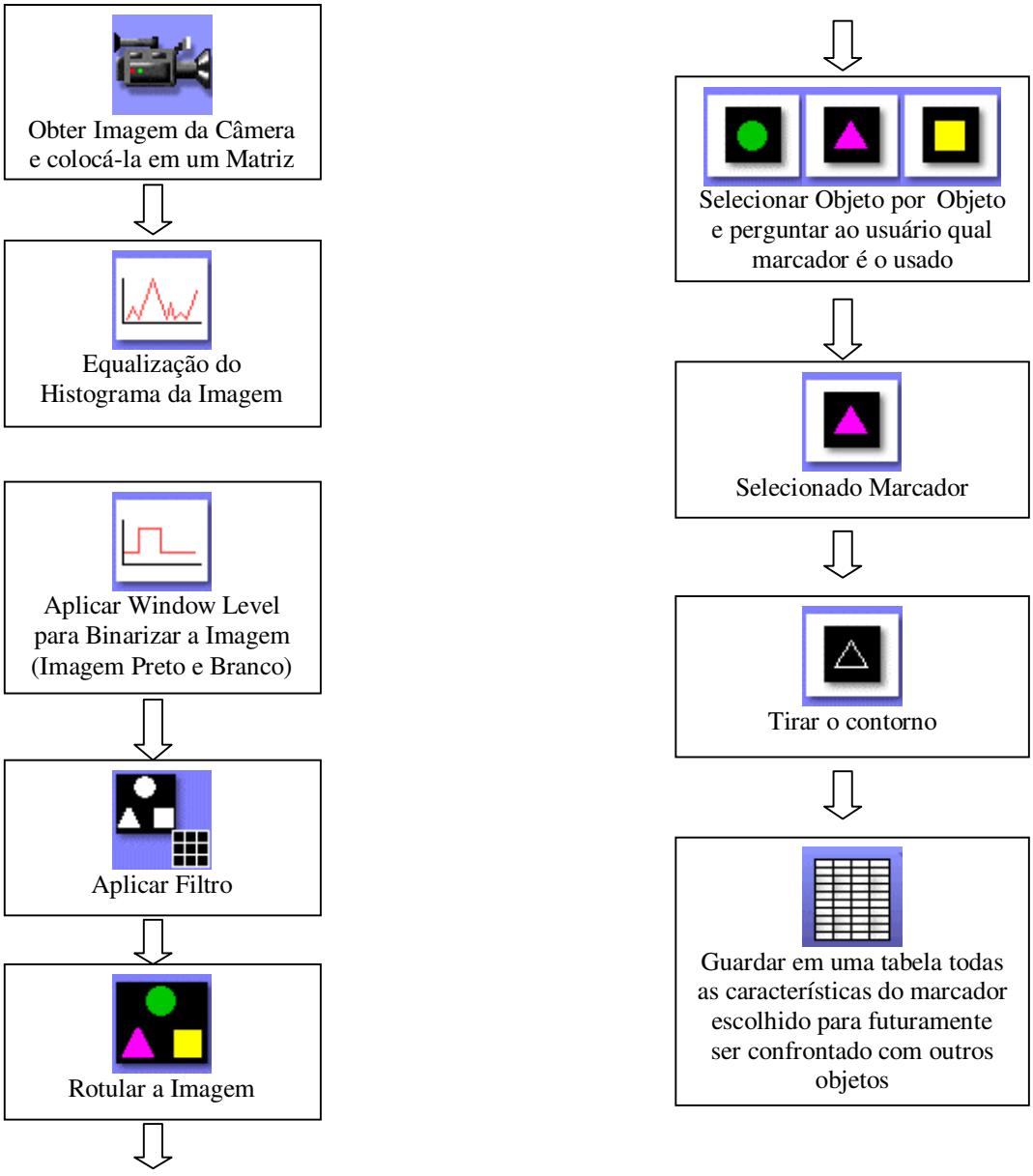
000555555555550070 – A posição inicial é zero, mouse parado. Logo em seguida, deslocamento do mouse para a direita. Depois de alguns ciclos, pára o mouse e dá um duplo click (070).

000300555555557777777700 – A posição inicial é zero, mouse parado. Após alguns ciclos, emite o sinal de click do mouse (030). Logo em seguida, volta o ponteiro para o centro e depois desloca o mouse para a direita e para baixo. No final, volta ao centro finalizando o movimento do mouse.

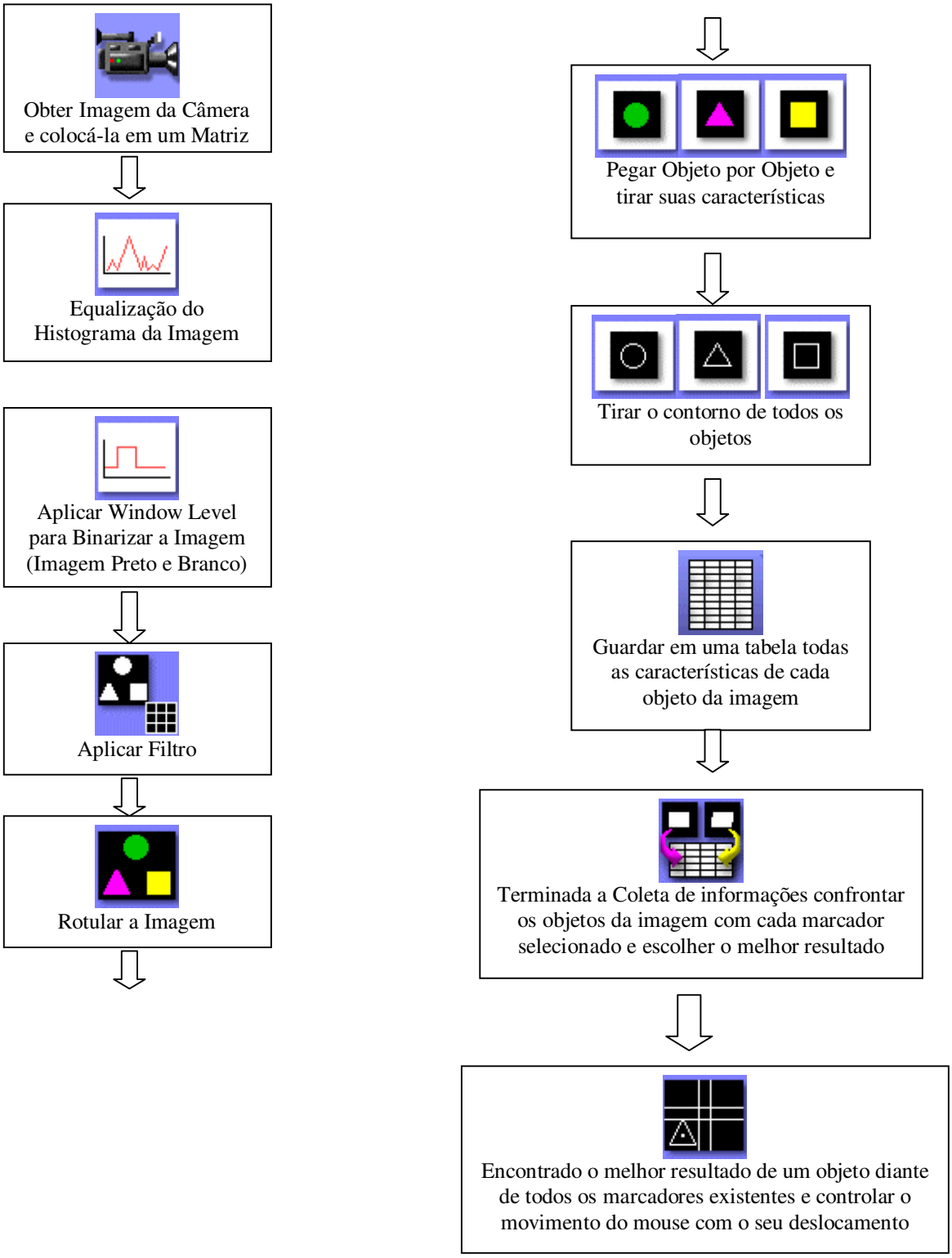
7. Seqüência de Todo o Processo

Neste capítulo descreve-se, passo a passo, os processos de calibração e utilização do software.

7.1. Encontrando um Marcador



7.2. Executando o software REMMC



8. Testes do Sistema

Os experimentos foram realizados com adultos e crianças normais e também com adultos e crianças com alguma deficiência motora. Alguns dos testes foram realizados para comprovar a sua eficiência quanto à variação de luminosidade, variabilidade do marcador no decorrer do uso e a resistência da localização do marcador em um ambiente com vários objetos parecidos. Esses testes ajudaram a comprovar a eficiência e facilidade da operação do software.

8.1. Posicionamento da Câmera

Foram realizados vários testes utilizando a câmera em diversas posições. Foi através destes experimentos que se criou a opção de inversão da imagem nos eixos X e Y (5.1.2 - Inversão da Imagem), com a finalidade de ajustar as possíveis mudanças de referência com a mudança de posição da câmera.

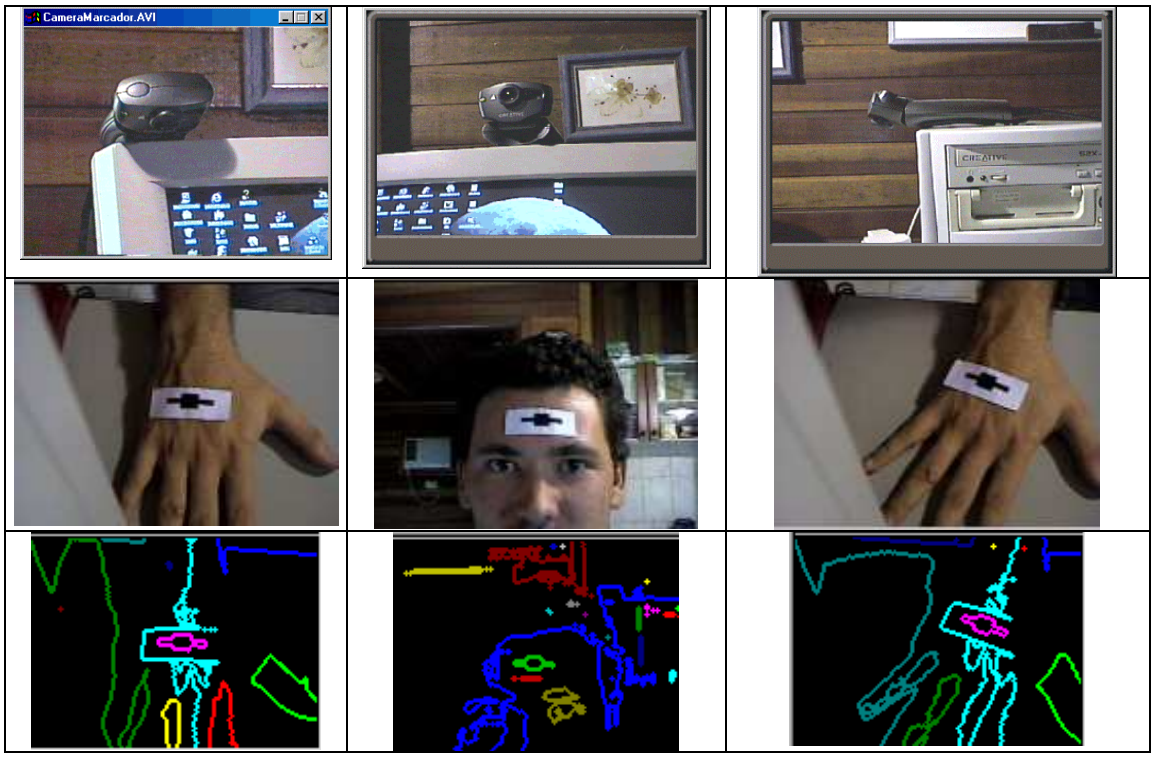


Fig. 8-1 - Exemplo de Posicionamento da Câmera

8.2. Colocação do Marcador

O marcador pode ser colocado em qualquer parte do corpo sendo que, o próprio usuário poderá definir o melhor local para colocá-lo. É preferível e aconselhável que o marcador seja colocado em alguma parte do corpo sobre a qual a pessoa tenha o melhor controle motor para que seja capaz de realizar os movimentos.

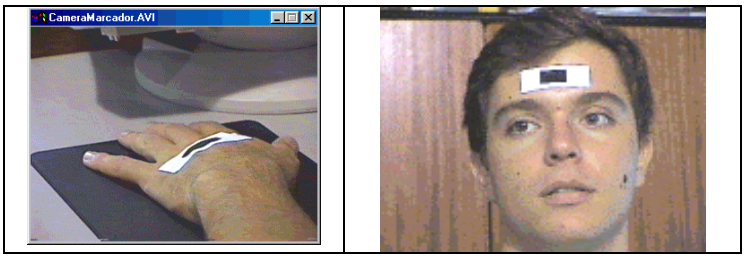


Fig. 8-2 - Colocando um Marcador

8.3. Calibração do Marcador

Na calibração o software obtém as características do objeto marcador, essas informações são passadas ao programa na fase inicial, antes de executar o reconhecimento do sistema. Para melhorar o reconhecimento do marcador, são passadas ao sistema diversas posições do marcador para que haja uma redundância do seu formato.

Quanto mais modelos o sistema adquirir do formato do marcador, maior será a qualidade e confiança do seu reconhecimento, porque terá uma grande quantidade de modelos a serem comparados. A única preocupação, neste caso, é com o comprometimento da velocidade do sistema, porque cada modelo será analisado com todos os objetos da imagem, sobrecarregando o processamento pelo computador.

8.3.1. Marcador em Várias Posições

Salienta-se que este é o modo mais correto para utilizar o sistema e o reconhecimento preliminar do marcador, em diversas posições e em alguns ângulos diferentes:

Exemplo:

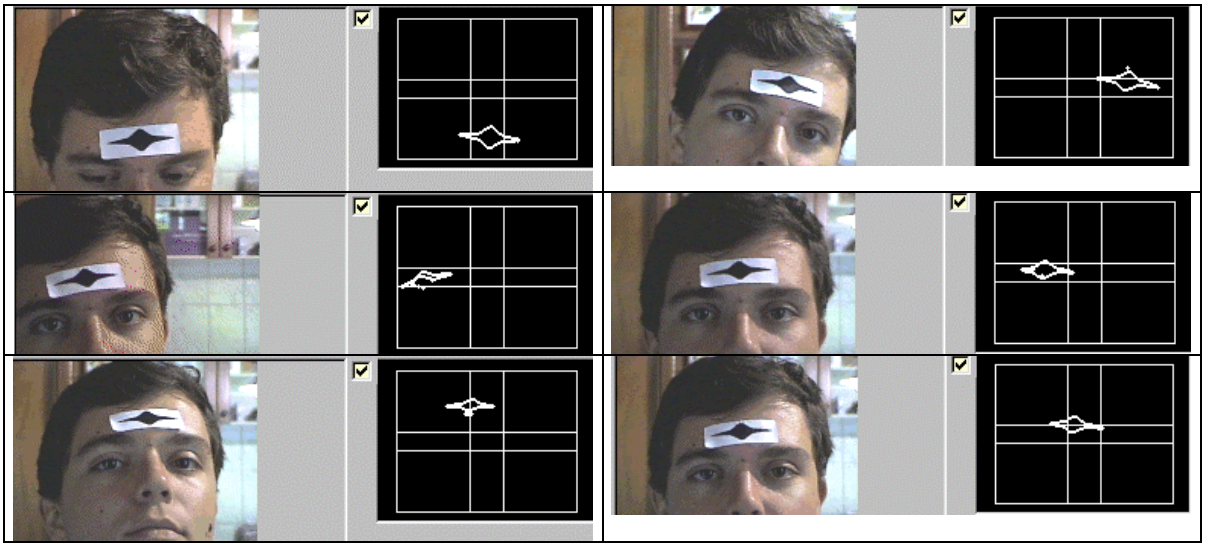


Fig. 8-3 - Selecionando o Marcador em Diversas Posições

Deve-se adquirir pelo menos um exemplo de marcador em cada uma das células de direção.

8.4. Treinos

Após ter calibrado o software e ajustadas todas as condições do ambiente de uso, o usuário precisará fazer um treinamento junto ao programa para se adaptar à nova forma de operar o computador, e aumentar sua capacidade de controle do movimento do marcador. O tempo para a nova adaptação não é estipulável, tendo em vista que varia de pessoa para pessoa.

8.5. Robustez

Foram feitos alguns testes para comprovar a robustez em diversos ambientes de uso. A variação de luminosidade e a deformação do marcador no decorrer do uso do software foram os maiores problemas encontrados no desenvolvimento do projeto.

8.5.1. Resistência à Variação de Luminosidade

O software foi testado mudando-se o referencial de luz para impor variação de luminosidade num mesmo ambiente, mesmo depois de calibrado.

	Calibração normal para o ambiente em uso
	Colocando uma luz mais forte do lado esquerdo do monitor

	Colocando uma luz mais forte atrás do usuário
	Colocando uma luz mais forte do lado esquerdo e direito do monitor
	Colocando uma luz mais forte do lado esquerdo do monitor e uma outra atrás do usuário.

Fig. 8-4 - Verificando a Robustez do Software

Em todas as situações anteriores, o sistema respondeu bem aos movimentos. O problema observado corresponde ao tempo de reajuste de brilho e contraste da câmera de vídeo, que não é imediato. Nesse período, o sistema pára de responder ou encontra um outro objeto similar ao marcador.

8.5.2. Deformação do objeto Marcador por Movimentação da Câmera

Dependendo do movimento que o usuário fizer com o marcador, este poderá sofrer alguma deformação na sua forma ou em seus ângulos. As funções de pesquisa levam em consideração essas mudanças para encontrar o objeto.

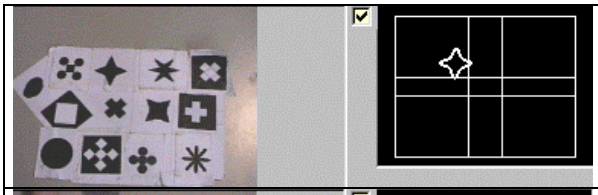
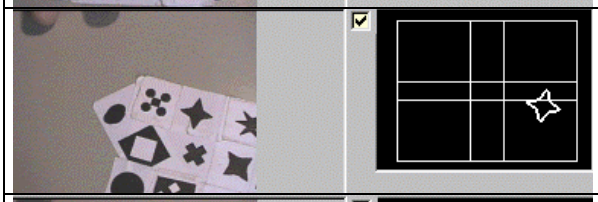
	Marcador na posição vertical
	Marcador levemente rodado para a direita
	Marcador deformando

Fig. 8-5 - Exemplos de Deformação do Marcador

8.5.3. Resistência a Vários Objetos no Ambiente

Os testes comprovaram uma certa resistência à variabilidade de objetos dentro de uma mesma imagem. No exemplo abaixo, o marcador escolhido foi filtrado, mesmo tendo várias outras figuras parecidas.

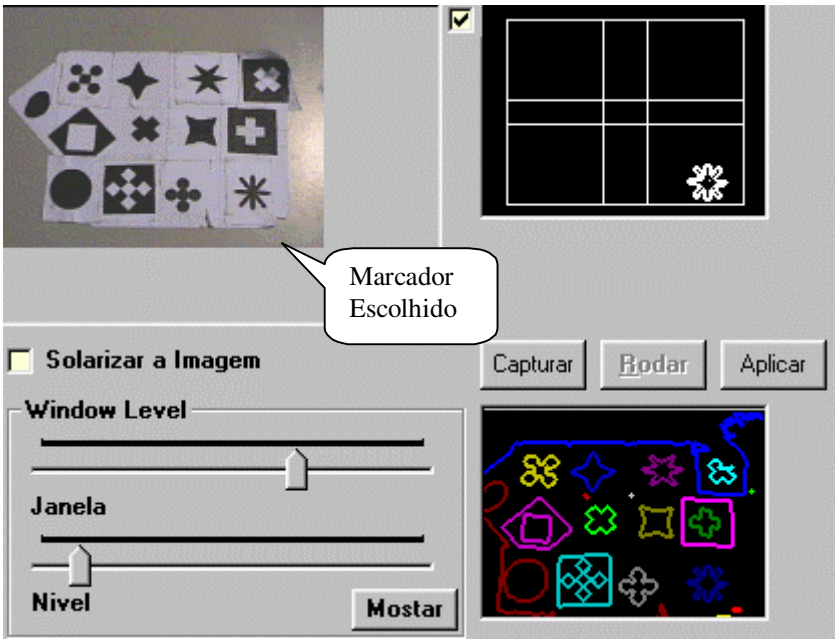


Fig. 8-6 - Buscando o Marcador no Meio de Objetos Parecidos

Estes testes também foram realizados com objetos distratores colocados no próprio ambiente de trabalho. Nessas condições, os marcadores também foram corretamente identificados.

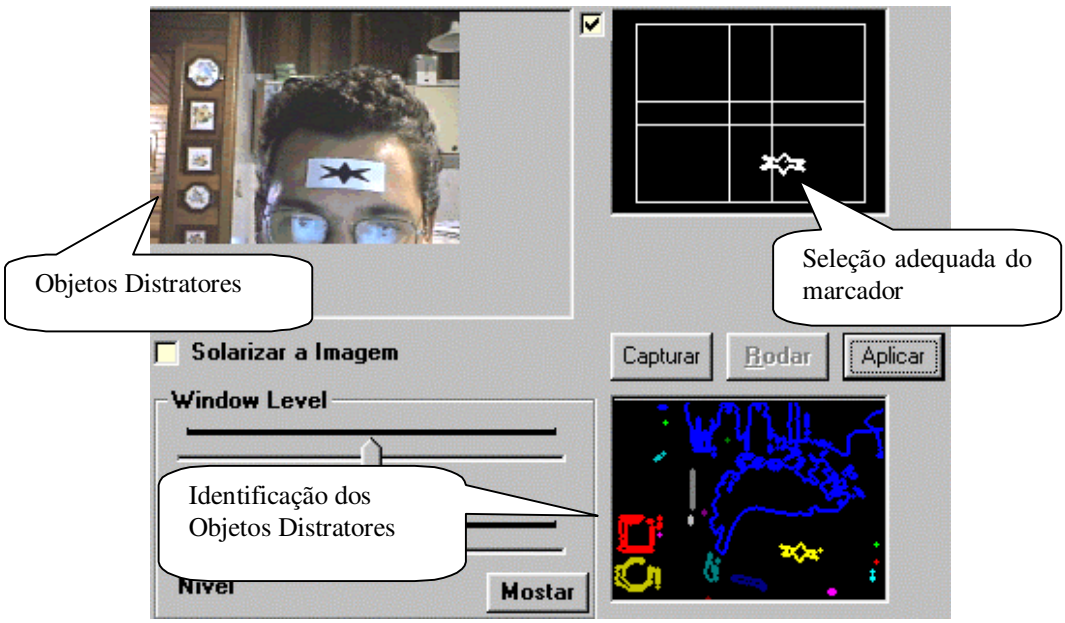


Fig. 8-7 - Reconhecimento do Marcador em Ambientes com Objetos Distratores

8.5.4. Reutilização de tabelas de Marcadores.

Com o sistema uma vez calibrado para um determinado ambiente e marcador, os dados podem ser salvos em arquivos, para posteriormente serem usados sem a necessidade de uma nova calibração.

Uma mesma calibração de um determinado marcador pode ser usada por uma ou mais pessoas num mesmo ambiente bem definido, que permita a configuração prévia de marcadores.

8.5.5. Testes em Computadores Diferentes

O sistema foi testado em diversos computadores e com vários modelos de câmera de vídeo. O micro de menor poder computacional testado foi um Pentium II 333 Mhz com 64 MB de memória e o de melhor configuração foi um Pentium III 700 Mhz com 128 MB de memória.

O REMMC teve um comportamento satisfatório no micro de menor capacidade, embora tenha sido necessário reduzir o número de exemplos de marcadores e também desabilitar a opção de exclusão de objetos.

O REMMC foi testado com três modelos distintos de câmeras da marca Creative, um modelo da marca Logitech e uma Câmera Profissional Sharp. O sistema funcionou satisfatoriamente com todas as câmeras, mas a facilidade de sua calibração variou com a velocidade e número de quadros que cada uma dessas máquinas gera por segundo e também com a qualidade da imagem adquirida.

Os modelos de câmera foram:

- Creative (WebCam 3(USB) , WebCam Go (USB), WebCam (Paralela))
- Logitech (USB)
- Sharp (Sharp ViewCam, VL – E630)

8.6. Utilização do Software

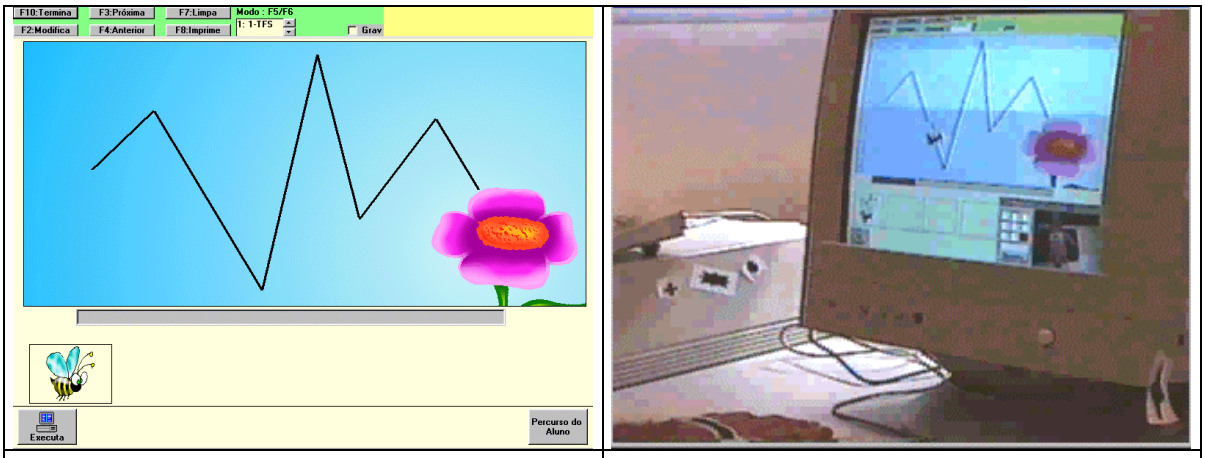
O sistema foi utilizado por um grupo de voluntários constituído de 7 adultos e 3 crianças sem problemas de coordenação motora, uma criança com pequena dificuldade de coordenação viso-motora, e outra com controle motor muito comprometido.

8.6.1. Adultos e crianças normais

Os adultos e crianças não mostraram nenhuma dificuldade de adaptação ao uso do sistema e conseguiram um controle preciso do mouse, sendo capazes de selecionar diversos utilitários através de seus ícones no ambiente Windows®.

Um dos utilitários acionados dessa maneira foi o sistema ENSCER® (desenvolvido com o apoio concedido pela FAPESP através do Programa "Inovação Tecnológica em Pequena Empresa"), que é um software educativo. Este sistema contém vários jogos para treinamento da coordenação motora. Nos testes realizados com os adultos, observou-se que todos foram capazes de controlar o software adequadamente, e executar os movimentos de mouse esperados.

Observou-se, ainda, que o desempenho dos indivíduos melhora com a utilização do sistema, demonstrando assim um efeito de aprendizagem que deve ser sempre utilizado para aumentar a produtividade do sistema.



Tela Original no Enscer®	Sistema sendo controlado com o REMMC

Fig. 8-8 - Testes do Software em Adultos e Crianças Normais

8.6.2. Crianças com problemas

A capacidade inicial de utilização do REMMC por crianças com deficiências foi bastante reduzida e diretamente relacionada com o grau de disfunção apresentada pela criança.

O indivíduo com melhor controle motor foi capaz, já na primeira sessão, de movimentar o mouse na tela do computador, mas foi incapaz de acionar qualquer aplicativo, por sua dificuldade de gerar as palavras de código curto **070** para simular o duplo click. O treinamento melhorou o controle do mouse, mas em função do pouco tempo trabalhado com essa criança, ainda não foi possível gerar, consistentemente, palavras curtas.

Os resultados obtidos com as crianças deficientes foram muito iniciais, mas chamou a atenção para alguns aspectos relacionados à motivação e treinamento. Em um dos testes realizados, a criança só é capaz de realizar movimentos grosseiros e abruptos, o que dificultou a calibração dos marcadores. Após 30 minutos de experimentação, conseguiu-se realizar algumas tarefas simples e em alguns momentos a criança foi capaz de modificar a marca na tela minimizada, que traduz a movimentação realizada. Esse resultado foi acompanhado de manifestação de alegria da criança e um aumento no interesse pela realização da tarefa. Outra dificuldade no uso do REMMC com essas crianças foi a qualidade da câmera de vídeo Logitech que é muito baixa.

Acredita-se que a programação do sistema para detecção de um número menor de movimentos e mais bruscos deva ser utilizada para uma fase de treinamento que aproveite o interesse da criança em aprender o controle da ferramenta. À medida que o treinamento contribuir para a aquisição de um melhor controle motor, os objetivos iniciais do sistema (acionamento do mouse com propósito de controle do computador) poderão ser retomados. No momento, parece claro que o REMMC pode ser utilizado por essa criança como uma ferramenta para treino de suas habilidades motoras.

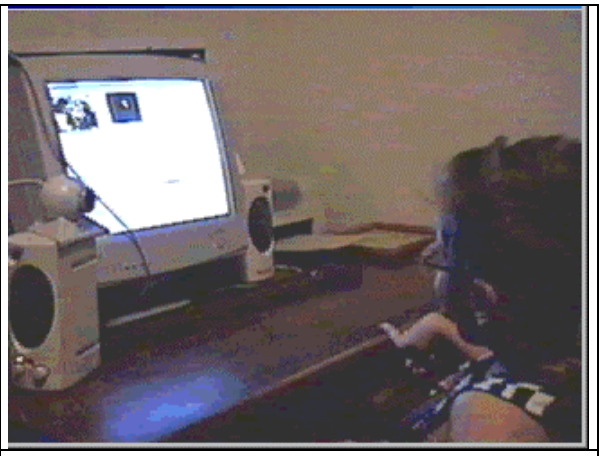
	
A criança com o Marcador	A criança controlando o movimento do marcador

Fig. 8-9 - Teste com Criança Portadora de Deficiência Mental e Física

9. Conclusões

O REMMC foi desenvolvido dentro da arquitetura teórica proposta por *Rocha* (1997), que procura simular o funcionamento cerebral para o processamento visual.

A imagem na retina sofre um tratamento básico de filtragem, onde contrastes são aumentados, as cores básicas e a luminância são identificadas. Nas áreas talâmicas, oponências de cores e refinamento de contrastes são processados, e principalmente as informações provenientes dos dois olhos são integradas, com o mesmo objetivo (*Clay Reid, 1999; Joseph, 1996; Rocha, 1999*).

As áreas corticais visuais iniciais estão encarregadas de isolar elementos conexos, descontinuidades etc., bem como identificar as características básicas desses elementos, tais como contorno, direções preferenciais, angulações (pontos significativos) etc. (*Ishai et al, 2000, Kanwisher et al, 1997; Kanwisher and Wojciulik; 2000; Kosslyn et al, 1999; Nakamura et al, 2000*).

Essas informações são utilizadas por áreas associativas visuais para identificação de características globais dos elementos isolados, que permitem reconhecer o objeto a que estão associados. Para tanto, cálculos adicionais de tamanho, rotação, centróide etc. são realizados por áreas que possuem campos visuais extensos, e que têm por finalidade extrair e comparar características invariantes de objetos (*Maunsell and Ferrera, 1995; O'Craven and Kanwisher, 2000; Rocha, 1999; Ungerleider, Coutney and Haxby; 1998*).

O reconhecimento do objeto é feito a partir da coerência entre as informações geradas pelas áreas associativas e de sua comparação com parâmetros anteriormente aprendidos como característicos de um objeto. Esse processo de comparação é baseado em um cálculo aproximado (fuzzy), para cálculo de semelhanças. O reconhecimento final de um elemento isolado é definido por um processo binário, em que o objeto que será a ele associado é o que possui características mais semelhantes àquelas atuais do elemento em análise.

O reconhecimento visual depende da ação de um conjunto grande de neurônios distribuídos por várias áreas corticais e principalmente das relações entre as informações fornecidas por cada um desses neurônios. É, portanto, o resultado da atividade de um sistema de processamento distribuído fracamente hierarquizado (*Rocha, 2000, Ungerleider, Coutney and Haxby; 1998; Young, 1995*), que se convencionou chamar de *what pathway* ***(via O Que)***. Finalmente, a localização e a possível movimentação do objeto são finalmente identificadas com a participação de neurônios distribuídos ao longo da *where pathway* ***(via Onde)*** e de neurônios especializados na análise de movimento.

O REMMC simula esse tipo de processamento, uma vez que a imagem adquirida é inicialmente submetida a vários processos de filtragem que procuram simular o tratamento visual no nível da retina e tálamo. Essa imagem filtrada é, então, analisada por várias rotinas que procuram isolar elementos a partir de sua conectividade, identificar direções preferenciais de seu contorno, seus pontos significativos, sua centróide, relações entre pontos do contorno (inicial e final) e entre eles e o centróide. Todas essas informações são utilizadas para memorização dos exemplos de marcadores durante a fase de reconhecimento e para sua identificação na fase operacional.

A aquisição de cenas dinâmicas apresentam menor qualidade devido a distorções proporcionados pelo próprio movimento dos objetos, exigindo mais eficiência dos filtros criados em software (*Chellappa, 1995*). Outra característica adotada é o rastreamento do objeto marcador com base em uma imagem anterior com a finalidade de indicar a trajetória do objeto marcador dentro da imagem (*Feris et al., 2000; Campos et al., 2000*).

O reconhecimento do marcador está baseado em um cálculo semelhante e uma estratégia de decisão binária baseada na maximização do grau de similaridade. Essa estratégia se mostrou robusta e simples, gerando um aplicativo que atende às especificações definidas no projeto inicial (*Serapião, 1997; Rocha, 1997*). Além disso, o REMMC parece poder ser utilizado também como uma ferramenta para estudo do aprendizado motor, e sua estrutura básica poderá ser modificada para atender a outros propósitos, como por exemplo identificação de objetos em uma linha de produção, controle dessa mesma linha etc.

O REMMC permite uma fácil calibração para garantir uma boa separabilidade do marcador na identificação de movimentos. Os processos de filtragem, idealizados e utilizados, garantem uma robustez do sistema, principalmente no que se refere a variações da luminosidade ambiental. Dentro dos ambientes internos, nos quais foi possível testar o sistema, a identificação correta do marcador foi realizada em diversas condições de luminosidade e de variação intensa e brusca da mesma. Muitas dessas variações não serão encontradas nos ambientes habituais para os quais se propõe o uso do REMMC.

Não foi possível testar o sistema em ambientes externos, devido à indisponibilidade de microcomputadores portáteis e de suportes adequados para a câmera de vídeo. Porém, os dados até agora obtidos parecem indicar que a robustez do REMMC permitirá essas adaptações futuras.

O uso do REMMC é simples, adultos e crianças (e mesmo aquelas com deficiências) são capazes de iniciar imediatamente seu uso com um mínimo de instruções. Os indivíduos que possuem um controle normal da motricidade obtêm um bom desempenho já nos primeiros momentos de uso do sistema, enquanto que aqueles com problemas de coordenação motora poderão necessitar de treinamento, cuja estratégia e duração dependerá do tipo de distúrbio motor.

O treinamento melhora o desempenho de todos os indivíduos que utilizam o sistema. Essa é uma característica interessante do REMMC, pois permite que ele seja utilizado como uma plataforma para estudo da aprendizagem e do controle motor. Se ele for acoplado ao sistema de aquisição e processamento do EEG, como o desenvolvido em nosso laboratório dentro do projeto ENSCER® (FAPESP – Programa de Inovação em Pequena Empresa), permitirá estudos sofisticados da motricidade. Esse é um resultado não antecipado na proposta original.

Definiu-se um sistema de codificação de movimento e posição de marcador que pode gerar dois tipos de palavras de código:

Controle Geral (CC): que podem ser utilizadas para controle de teclado, e

Controle do Mouse (CM): com a finalidade de controle do movimento e eventos do mouse

As palavras CC tem a seguinte estrutura $\mathbf{0i^j0}$, **i ε (1, 2, 3, 4, 5, 6, 7, 8)**, onde o valor de j pode ser ajustado para garantir o número de bits necessários ao controle desejado. As palavras CM têm a seguinte estrutura $\mathbf{0i^n0}$, **i ε (1, 2, 3, 4, 5, 6, 7, 8), n>j**, que permite um controle da movimentação do mouse que tem a precisão definida na matriz de localização de marcadores, no presente ensaio, em 3 bits. A capacidade do código do REMMC estará sempre subordinada à capacidade de controle motor do usuário.

A possível extensão do uso do REMMC para controle do teclado é outro resultado não previsto na proposta inicial. A estrutura de palavras criadas permite uma permutação fácil entre os dois modos de controle, uma vez que é definida por parâmetro temporal.

O REMMC foi projetado para trabalhar em background, de modo que pode ser utilizado com sistema Windows® para controle de qualquer utilitário definido para esse ambiente. Os testes realizados utilizaram o sistema ENSCER®, que é um software educativo para apoio à Educação Infantil e ao Ensino Fundamental. O sucesso do REMMC em controlar o ENSCER® abre a utilização desse software para crianças PC (Paralisia Cerebral), que necessitariam de mouse especial para poderem controlar o microcomputador. Agora, isso será possível com a utilização de equipamento padrão de multimídia.

O uso do REMMC para controle de aplicativos do tipo do Word, por exemplo, poderá ser desenvolvido em breve, uma vez que a estrutura de código assim permitir. O que será necessário é um estudo de otimização do código tendo em vista a freqüência de uso das teclas. Uma primeira idéia poderá ser a utilização de palavras curtas para as teclas de controle tipo Enter, Espaço, Retrocesso etc., e das palavras mais longas para caracteres infreqüentes tais como #, &, etc.

A capacidade computacional do microcomputador e as características da câmera de vídeo influem no desempenho do REMMC. Computadores rápidos e câmeras sofisticadas tornam o sistema mais robusto e eficiente. Mas o sistema funciona bem, mesmo com os

equipamentos considerados, hoje, fora de linha (Pentium II 333 MHz e Câmeras tipo Logitech USB).

Concluímos que o objetivo do trabalho, de criar um sistema de baixo custo e alta portabilidade, foi atendido. O sistema pode ser utilizado por quaisquer pessoas com deficiências físicas distintas, desde portadores de tendinites até tetraplégicos, usando um computador comum com as configurações mínimas necessárias. Além disso notamos a possibilidade do sistema ser utilizado em seções de fisioterapia, ajudando o usuário no treinamento motor.

Isso significa que a tecnologia torna as coisas mais fáceis para as pessoas normais, mas para pessoas portadoras de deficiências, torna as coisas possíveis.

10. Referências

AICARDI, J. **Diseases of the nervous system in childhood**. Mac Keith Press, London, 1998.

AIELLO, L. C. **Brains and guts in human evolution: The expensive tissue hypothesis**. Braz. J. Genetics, n.20, pp. 141-148, 1997.

ALEGRE, L., ROCHA, A.F., MOROOKA, C. K. **Inteligent approach rod pumping problens**. SPE 26253, pp. 249-255, 1993a.

ALEGRE, L., MOROOKA, C. K., ROCHA, A.F. **Inteligent diagnostics of rod pumping problems.** SPE 26516, pp. 97-107, 1993b.

BARON, I. S., FENNELL E. B., VOELLER K. K. S. **Pediatric Neuropsychology in the Medical Setting.** Oxford University Press, Oxford and N. York, 1995

BARON-COHEN, S. **Mindblindness.** Bradford Books, The MIT Press, Cambridge, 1995

BATCHELOR, E. S. & DEAN R. S. **Pediatric Neuropsychology**. Allyn and Bacon, Boston, 1996

CAMPOS T. E., FERIS R. S., CESAR R. M. Jr. **O Reconhecimento Computacional de Pessoas.** 11st. SEMAC, IBILCE - UNESP, S. J. do Rio Preto - SP, Brazil, Nov 2000.

CAPOVILLA, F. C. et al. **Tecnologia em (re)habilitação cognitiva**. São Paulo, SP: Edunisc-Sociedade Brasileira de Neuropsicologia. vol 1, ISBN: 85-87121-01-4; e vol 2, ISBN: 85-87121-02-2, 1998, 2000

CHELLAPPA Rama, WILSON Charles L., SIROHEY Saad. **Human and machine recognition of faces: A survey.** Proceedings of the IEEE, v.83, n.5, pp. 703-740, May 1995.

CLAY REID, R. **Vision in Fundamental Neuroscience** (Zigmond, Bloom, Landis, Roberts, Squire, eds.) Academic Press, 1999

DEFRIES, J. C. & GILLIS J. J. **Etilogy of reading deficits in learning disabilities: Quantitative genetic analysis**. In Developmental Neuropsychology, (Spreen, O., A. H. Risser and D. Edgell, eds.), Oxford University Press, Oxford and N. York, 1995

DUANE, D. D. *Biological foundations of learning.* **In: Developmental Neuropsychology**, (Spreen, O., A. H. Risser and D. Edgell, eds.), Oxford University Press, Oxford and N. York, 1995

FERIS R. S., CAMPOS T. E., CESAR R. M. Jr. **Detection and Tracking of Facial Features in Video Sequences.** Lecture Notes in Artificial Intelligence, v. 1793, p 129-137, Springer-Verlag press MICAI-2000, Acapulco, April 2000.

FREEMAN, H. ***Computer Processing of Line-Drawing Images.*** Computer Surveys, 6, pp. 57-97, 1974.

FREEMAN, H. ***On the Encoding fo Arbritary Geometric Configurations.*** IEEE Trans. Elec. Computares, v. EC-10, pp. 260-268, 1961.

GONZALES, R.C. & FITTES, B.A.. **Gray-Level Transformations for Interactive Image Enhancement.** Mechanism and Machine Theory, v. 12, pp. 111-122, 1977.

GONZALEZ, Rafael & WOODS, Richard. **Digital Image Processing.** Addison Wesley, 1992

HEIDRICH, Regina de Oliveira **Criatividade na Educação de Deficientes Mentais com o uso de Softwares Educativos.** Dissertação de Mestrado, UNESP, Bauru, 1999.

HUMMEL, RA. **Histogram Modification Techniques.** *Computer Graphics and Image Processing*, v. 4, pp. 209-224, 1975

ISHAI, A., UNGERLEIDER L. G., MARTIN A., & HAXBY J. V. **The representation of objects in the human occipital and temporal cortex.** J. Cognitive Neurosci. v.2, pp.35-51, 2000

JOSEPH, R. **The Occiptal Lobes, in Neuropsychiatry, Neuropsychology and Clinical Neuroscience.** Willians & Wilkins, pp. 471, Baltimore Philadelphia, 1996

KANWISHER, N. & WOJCIULIK E. **Visual Attention; Insights from brain imaging.** Nature Reviews: Neuroscience, v.1, pp. 91-100, 2000

KANWISHER, N., WOODS R. P., IACOBONI M. & MAZZIOTTA J. C. **A locus in human estrastriate cortex of visual shape analysis.** J. Cognitive Neurosci., v.9, pp. 133-142, 1997

KITTLER, J. & IiNGWORTH, J. **Minumum error thresholding.** *Pattern Recognition*, v. 19, no. 1, pp. 41-47, 1986

KOSSLYN, S. M., A. PASCUAL-LEONE, O. FELICIAN, S. CAMPOSANO, J. P. KEENAN, W. L. THOMPSON, G. GANIS, K. E. SUKELAND N. M. ALPERT **The role of area 17 in visual imagery:** Convergent evidence from PET and rTMS. Sicence, v. 284, pp.167-170, 1999

KURITA, T., OTSU, N., ABDELMALEK, N. **Maximum lkehood thresholding based on population mixture models.** *Pattern Recognition*, v. 25, no. 10, pp. 1231-1240, 1992

LACEFIELD, W. & GARTHEE, S. **Win-SCAN-DEMO.** Limited Edition, version 1.1 Lexington, KY, Academec Software Inc., 1995.

LARMAN, CRAIG **Utilizando UML e Padrões: Uma introdução à análise e ao projeto orientados a objetos**, Porto Alegre, Bookman, 2000, ISBN 0-13-748880-7

LITTLEFIELD B. & HANSELMAN D. **MATLAB 5 - Versão do Estudante - Guia do Usuário**. Makron Books do Brasil. 1999.

LUBS, H. A., RABIN M., CARLAND-SAUCIRE K., WEN X. L., GROSS-GELNN K., DUARA R., LEVIN B. & LUBS M. L. **Genetic bases of developmental dyslexia**: Molecular studies. In Developmental Neuropsychology, (Spreen, O., A. H. Risser and D. Edgell, eds.), Oxford University Press, Oxford and N. York, 1995

MANTOAN, M. T., VALENTE, J. A. **Contribuições para uma abordagem inovadora da educação de deficientes.** In: Mantoan, M. T., Ser ou estar: eis a questão - explicando do déficit intelectual . Rio de Janeiro: WVA Editora e Distribuidora., p.155-174, 1997

MARR, D., & HILDRETH, E. **Theory of Edge Detection.** Proc. R. Soc. Lond., v. B207, pp. 187-217, 1980.

MAUNSELL, J. H. R. & FERRERA V. P. **Attentional mechanisms in visual cortex.** In: The Cognitive Neuroscience, M. S. Gazzaniga (Ed), The MIT Press, Cambridge, pp.451-461, 1995

NAKAMURA, K., R. KAWASHIMA, N. SATO, A. NAKAMURA, M. SUGIURA, T. KATO, K. HATNAO, K. ITO, H. FUKUDA, T. SHCORMANN & K. ZILLES **Functional delineation of the human occipito-temporal areas related to face and scene processing.** Brain, v.123, pp.1903-1912, 2000

O'CRAVEN, K. M. & KANWISHER N. **Mental imagery of faces and places activates corresponding sitmulus-specific brain regions.** J. Cognitive Neurosci., v.12, pp. 1013-1023, 2000.

ORNSTEIN, R. **The evolution of consciousness,** A Touchstone Book, Simon & Schuster, N. York, 1991

OTSU, N. **A threshold selection method from gray level histograms.** *Pattern Recognition*, v. 9, no. 1, pp. 62-66, 1979

PETZOLD, Charles & YAO, Paul. **Programando para Windows 95.** Editora Makron Books do Brasil, 1996

PETZOLD, Charles, **Programming Windows.** 5. ed. Microsoft Press, 1998

ROCHA, A. F., **O cérebro na escola.** EINA, Jundiaí, 2000

ROCHA, A. F., **O cérebro: Um breve relato de sua função.** EINA, Jundiaí, 1999

ROCHA, A. F., A. B. SERAPIÃO; A. RONDÓ, E. C. RODELLA; F. LUCHINI; M. P. REBELLO, **Correlating Intelligence and Brain Activity.** Proc. 6th Ann. Meeting Cognitive Neurosciences Society, Washington, D.C, USA, 1999.

ROCHA, A. F., et al, **Computer diagnosing and teaching of the brain impaired children.** Proc. 5th Ann. Meeting Cognitive Neurosciences Society, San Fracnisco, USA, 1998.

ROCHA, A. F., **The brain as a symbol processing machine.** *Progress in Neurobiology*. v. 53, no.2, pp. 121-198, 1997

ROCHA, A. F., et al **ENSCER – um sistema informatizado para ensino e diagnóstico de deficientes cerebrais**. Processo 97/6020-9, PIPE – FAPESP, 1997.

RUMBAUGH, J., BLAHA, M., PREMERLANI, W., EDDY, F., LORENSEN, W., **Modelagem e Projetos Baseados em Objetos.** Editora Campos, 8 ed., 1994.

SANTAROSA, L. M. C., MARTINS, A. R., SILVEIRA, M. S., FRANCO, B. S. **Adaptação para o Português e Avaliação de um simulador de Teclado para portadores de Paralisia Cerebral.** In: Anais do II Congresso Ibero-americano de Informática na Educação. v.1, pp.116-118, Portugal, 1994a.

SANTAROSA, L. M. C., SILVEIRA, M. S., VIRTI, D. S. **Hipermeios na Construção da Leitura e Escrita: ambiente para Crianças e Portadores de Deficiências.** In: Anais do II Congresso Ibero-americano de Informática na Educação. v.1, pp. 119-121, Portugal, 1994b

SCHILDT, H. ***C:- Completo e Total.*** São Paulo: Makron/McGraw-Hill, 3. ed. 1997

SERAPIÃO, A. B. S. ,**Sensor: An Artificial Visual System.** Progress Neurobiology, v. 53, no.2, pp.179-183, 1997.

SERAPIÃO, A. B. S., ROCHA, A. F., REBELO, M. P. F.. **Toward a Theory of Genetic Computing.** In Genetic Algorithms and Soft Computing, F. Herrera and J. L. Verdegay, (Eds) (Physica Verlag, Heidelberg); pp. 68-94, 1996

SPREEN, O., RISSER A. H. & EDGELL D. **Developmental Neuropsychology.** Oxford University Press, Oxford and N. York, 1995

UNGERLEIDER, L. G., COUTNEY, S. M. and HAXBY , J V. "**A neural system for human visual working memory."** Proc. Natl. Acad. Sci., 95: 883-890, 1998

VALENTE, J.A. **"Funções executivas na criança com déficit de atenção: avaliação utilizando testes neuropsicológicos e atividades de programação em Logo."** Dissertação de Doutorado, Faculdade de Ciências Médicas da Unicamp, Campinas, 1998

VALENTE, J. A **"Informática na Educação Especial.**", Anais do III Congresso Brasileiro sobre Educação Especial. Curitiba. pp. 9-25, 1999

WOODS, R.E. & GONZALES, R.C. **"Real Time Digital Image Enhancement."** Proc. IEEE, v. 69, n.5, pp. 643-654, 1981.

YOUNG, M. P. "**Open questions about the neural mechanisms of visual pattern recognition.**" In The Cognitive Neuroscience, M. S. Gazzaniga (Ed), The MIT Press, Cambridge, pp.463-474, 1995