ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

10,0 SEIS E ZERO

ABNER DE RINALDI TRAVERSIM OLIVA
MARCELODE MELO ABDO GANEU

- PROCESSAMENTO DE IMAGENS -

SISTEMA DE DETECÇÃO DE FORMAS GEOMÉTRICAS SIMPLES EM IMAGENS DIGITAIS

PROF. ORIENT.: NEWTON MARUYAMA

SÃO PAULO
2000

Às nossas famílias
e a todos que de alguma
forma colaboraram
com esse trabalho de formatura

RESUMO

A automação através de sistemas computadorizados é algo que pode ser encontrado hoje em dia em todos os setores produtivos e serviços a eles relacionados, tais como segurança, conforto térmico, toda a sorte de controles e seus respectivos procedimentos para a manutenção e/ou restabelecimento das condições desejadas ou aceitáveis, com grande racionalização dos recursos que devem ser empregados em tais atividades, tornando-se na maioria dos casos uma condição *sine qua non* para se atingir os níveis de produtividade e confiabilidade necessários em um mundo grandemente competitivo.

Muitos desses sistemas se utilizam de imagens para desempenhar suas funções, seja um aparentemente simples sistema de vigilância de pessoas ou de um complexo sistema produtivo.

Os dispositivos utilizados para a obtenção da imagem desejada são genericamente chamados de *câmeras* e em uma analogia quase perfeita com os seres vivos é o olho do sistema, pois o olho apenas capta a imagem, que é processada pelo cérebro, sendo de atribuição da câmera à obtenção de uma imagem de qualidade suficientemente boa para que o sistema possa dela tirar as informações necessárias. Ainda na analogia com os olhos dos seres vivos, as câmeras possuem a grande vantagem de não precisarem estar fisicamente ligadas ao sistema.

Alguns exemplos de sistemas que utilizam câmeras como "olhos":

- Sistemas operatórios onde os médicos podem ver e operar o corpo humano, sendo necessários pequenas incisões suficientes para a introdução de finos cateteres.

- Robôs que realizam tarefas perigosas, insalubres ou onde a presença humana não é possível.

- Sistemas produtivos onde é necessária a monitoração e/ou observação de ambientes como os citados no item anterior, tais como fornos, misturadores, equipamentos que estão a uma grande distância da sala de controle, etc.

ABSTRACT

The automation through computerized systems is something that can be found nowadays in all the productive sectors and jobs related to them, such as security, thermal comfort, all the luck of controls and its respective procedures for the maintenance and/or reestablishment of the desired or acceptable conditions, with great rationalization of the features that must be used in such activities, becoming in the major of the cases a condition *sine qua non* to reach the levels of necessary productivity and reliability in a greatly competitive world.

Many of these systems if use pictures to play its functions, either parentally a simple system of monitoring of people or a complex productive system. The devices used for the attainment of the desired picture generically are called cameras and in an almost perfect analogy with the beings livings creature it is the eye of the system, therefore the eye only picks-up the picture, that is processed for the brain, being of attribution of the camera to the attainment of a picture of enough good quality so that the system can of it take off the information necessary.

Still in the analogy with the eyes of the beings livings creature, the cameras possess the great advantage not to need to be physically on to the system. Some examples of systems that use cameras as " eyes ":

- Operational Systems where the doctors can see and operate the human body, being necessary small enough incisions for the introduction of fine catheters
- Robots that do dangerous tasks carry through, unhealthy or where the presence human being productive Systems are not possible

- where it is necessary the monitoring and/or comment of environments as the cited ones in the previous item, such as ovens, mixers, equipment that they are to a great pitch of the control room, etc

SUMÁRIO

INTRODUÇÃO

O sistema que será desenvolvido objetiva a detecção e identificação de objetos novos a uma dada cena. A motivação para este trabalho está na crescente utilização de imagens e seu respectivo processamento nas diversas áreas de engenharia, notadamente nos processos automatizados, como por exemplo no controle de qualidade em chão de fábrica.

Sendo assim, o trabalho objetiva a criação de um sistema que faça a identificação automática de uma mudança em um ambiente que está sendo monitorado, de modo a evitar falhas humanas nessa tarefa, e também identificar alguns padrões geométricos pré-definidos conseguindo-se dessa maneira identificar com alguma aproximação qual o objeto que está sendo analisado.

O sistema que será desenvolvido deve ser capaz de quantificar a dessemelhança entre a imagem de referência e a imagem atual, decidindo se essa dessemelhança é suficiente para que possa-se considerar que existe um objeto novo à cena.

Sabe-se que a utilização e o processamento de imagens ainda é uma técnica em desenvolvimento e que carece de muitas pesquisas e também desenvolvimento de novas tecnologias, principalmente em *hardware*, cabendo-se destacar a exigência de elevada capacidade de processamento necessária. Quando se trabalha com imagens, sempre há o compromisso entre a precisão necessária

para a tarefa e a capacidade do sistema que a realiza. Muitas vezes esse compromisso é difícil de ser obtido, razão pela qual métodos que envolvem processamento de imagens não podem ser aplicados a diversos segmentos da indústria.

Uma vez que há a necessidade de haver o compromisso anteriormente exposto, definiremos o nível de precisão que será utilizada neste trabalho, dada a capacidade do hardware existente e que foi utilizado ao longo deste trabalho.

Algumas premissas foram definidas de modo a não inviabilizar o bom andamento deste trabalho, pelo nível que complexidade que seria necessário para atender uma situação totalmente genérica. Deve-se destacar que esse trabalho não tem a intenção de abranger todos os casos possíveis, como a maioria dos sistemas existentes.

As premissas que foram adotadas ao longo deste trabalho foram:

- Identificação de figuras geométricas simples;
- Figuras não côncavas, ou seja, sem reentrâncias;
- Figuras não furadas;
- Na área de trabalho não pode haver mais que uma figura;

Para a otimização do uso do sistema pronto também é muito importante que haja uma parametrização onde o usuário possa adequar o sistema às suas necessidades.

Além de identificar a forma, seria interessante que o sistema fornecesse informações sobre as figuras identificadas, tais como baricentro, vértices, raio (no caso de círculos), etc., pois essas informações podem ser úteis a outros sistemas.

Com o desenvolvimento da tecnologia, a capacidade de processamento e armazenamento de imagens está se tornando mais eficiente possibilitando sua utilização com maior freqüência na indústria de um modo geral. Isso é uma tendência verificada nos últimos anos contribuindo para o aumento da eficiência na produção, o que nos motivou a desenvolver este trabalho que abordará a obtenção, detecção e identificação de imagens.

REVISÃO DA LITERATURA

Metodologia de Projeto

Para o desenvolvimento de um projeto, existem muitas metodologias que podem ser aplicadas, não se podendo afirmar que uma é certa ou melhor do que outra comparando-se metodologias entre si, já que uma metodologia pode aplicar-se melhor a um tipo de projeto e outra a outro. O que pode-se afirmar é que a metodologia a ser adotada dependerá de uma análise detalhada do projeto, levando-se em conta todos os requisitos que envolvem o projeto em si.

Os projetos podem ser diferenciados entre si quanto à evolução em *projeto evolutivo*, caracterizado por mudanças relativamente lentas, acompanhando as exigências de mercado, e o *projeto inovador*, que é caracterizado por soluções radicalmente novas, que utilizam as últimas descobertas técnico-científicas. Quanto à abordagem, existem os *projetos por normas*, que caracterizam pela utilização de normas ou regras já estabelecidas para certos requisitos de projeto, método de cálculo, etc., e os *projetos racionais*, onde existe a abordagem específica para cada problema, com o desenvolvimento de modelos com base técnico-científica. A confiabilidade dos resultados dependerá da qualidade dos modelos utilizados, e pode exigir ensaios para a sua validação.

Apesar das especificidades já citadas, existem determinadas características que são comuns a todos os projetos. Uma característica básica a todos os projetos de engenharia é que o seu desenvolvimento não se dá de uma forma linear, com cada item sendo completamente definido antes de se passar para o seguinte. Uma vez que existe uma interdependência entre os diversos itens de um projeto, o seu desenvolvimento é interativo, sendo que partindo de uma definição mais grosseira vai-se aprimorando cada vez mais cada um dos itens até obter-se a configuração desejada. De acordo com [1], os projetos de engenharia apresentam as seguintes características genéricas, antes, durante e após a sua execução:

- Necessidade: O projeto deve ser resposta ou solução a uma necessidade individual ou coletiva.

- Exequibilidade física: O projeto deve ser um produto ou processo fisicamente realizável

- Valor econômico: O produto ou processo desenvolvido pelo projeto devem ter para o consumidor uma utilidade igual ou maior à soma de todos os custos envolvidos em pô-lo a sua disposição. Deve ser compensador para o seu fabricante ou executor. O valor econômico é a relação entre o valor atribuído ao produto e o valor a ser pago por ele.

- Viabilidade financeira: Os custos de projeto, produção, distribuição de um produto devem ser financeiramente suportáveis pela instituição executora.

- Otimização: A escolha final de um projeto deve ser a melhor entre as várias alternativas; cada característica do projeto deve ser a melhor possível.
- Critério de projeto: A otimização deve ser feita de acordo com um critério que represente o equilíbrio a ser conseguido pelo projetista entre vários requisitos, em geral conflitantes. Incluem-se as exigências e expectativas do consumidor, fabricante, distribuidor e da sociedade como um todo.

- Subproblemas: No desenvolvimento de um projeto continuamente aparecem subproblemas de cuja solução depende a solução do problema global.

- Aumento da confiança: O projeto é uma atividade em que os conhecimentos produzidos durante o processo permitem a transição da incerteza para a certeza do sucesso de um produto. A cada etapa a confiança no projeto deve aumentar.

- Custo da certeza: O custo das atividades destinadas à obtenção de conhecimentos sobre o projeto deve corresponder proporcionalmente ao aumento da certeza sobre o sucesso.

- Bases para as decisões: Um projeto deve ser interrompido sempre que as informações disponíveis indiquem o seu fracasso; será continuado somente se as informações garantirem a conveniência da aplicação dos recursos necessários à fase seguinte.
- Requisitos Mínimos: Em qualquer fase do projeto as decisões que fixam as suas características devem ser as estritamente necessárias para a continuação imediata. Com isso evita-se que as soluções de futuros problemas sejam restringidas desnecessariamente.

- Apresentação: O projeto é em essência a descrição de um produto ou processo, normalmente apresentado na forma de documentos, relatórios, desenhos, maquetes e outros.

Imagens digitais

Matematicamente, uma imagem pode ser definida como uma função $I(x,y)$, definida numa certa região. Para ser possível a manipulação por computadores digitais é necessário que a imagem esteja no formato digital. Uma imagem digital (ou discreta) é aquela onde a função só é definida numa grade regular de pontos.

Uma imagem digital pode ser obtida de uma imagem não-digital (contínua) através de um processo que envolve dois passos: "amostragem", que consiste em

discretizar o domínio de definição da imagem e "quantização", que consiste em escolher um valor para a imagem em cada ponto. Assim, uma imagem digital pode ser vista como uma matriz com N linhas e M colunas.

BitMap

Formato BitMap:

Os arquivos bitmap são armazenados em um dispositivo especial chamado DIB (device independent bitmap) que permite o windows exibir o arquivo em qualquer dispositivo de vídeo. O termo device independent significa armazenamento de cores independente do método utilizado para representar a cor no vídeo. O nome definido para os arquivos bitmap é .BMP.

A estrutura do BitMap:

Cada bitmap contém um bitmap file header, bitmap information header, uma tabela de cores e um vetor que define os bits do bitmap. O arquivo é formado por:

```
BITMAPFILEHEADER bmfh;
BITMAPINFOHEADER bmih;
RGBQUAD          aColors[];
BYTE             aBitmapBits[];
```

O bitmap file header contém informações sobre o tipo, tamanho e forma do DIB, o bitmap information header contém informações sobre dimensões, tipo de

compressão e o formato de cor para cada bitmap. A tabela de cores contém tantos elementos quantas forem as cores do bitmap. As cores na tabela são armazenadas em ordem de importância, isso auxilia o dispositivo de vídeo a exibir as cores e melhoram a qualidade da imagem uma vez que alguns dispositivos não suportam todas as cores.

A estrutura BITMAPINFO pode ser usada para armazenar informações do bitmap information header e da tabela de cores. Os bits seguintes a tabela de cores correspondem a um vetor de Bytes representando as linhas (scan line) do bitmap. Cada scan line representa uma seqüência de bytes representando os pixels da linha. O tamanho da scan line depende das cores e da largura da figura em pixels. As scan lines são armazenadas de baixo para cima, ou seja, o primeiro byte representa o pixel mais baixo a esquerda e o último o pixel mais alto a direita.

Cada pixel definido pela estrutura bitmap information header pode ter os seguintes números de bits:

1 O bitmap é monocromático e a tabela de cores contém duas entradas. Cada bit no vetor representa um pixel. Caso o bit esteja em branco o pixel é mostrado com a cor da primeira entrada na tabela de cores e se o bit for setado o pixel é exibido com a cor da segunda entrada na tabela.

4 O bitmap tem no máximo 16 cores. Cada pixel é representado por 4 bits na tabela de cores. Por exemplo, se o primeiro byte no bitmap for 0x1F, o byte representa dois pixels. O primeiro pixel contém a cor da segunda entrada na tabela de cores e o segundo pixel contém a cor da décima sexta entrada na tabela de cores.

8 O bitmap contém no máximo 256 cores, cada pixel é representado por um byte na tabela de cores. Por exemplo, se o primeiro byte no bitmap for 0x1F o primeiro pixel terá a cor da trigésima segunda entrada na tabela.

24 O Bitmap tem no máximo 2^24 cores, cada 3 bytes no bitmap representa uma intensidade de vermelho, verde e azul para o pixel.

Compressão de BitMap:

Windows versão 3.0 ou posterior suporta formatos de compressão que utiliza 4 bits ou 8 bits por pixel. A compressão reduz o espaço de armazenamento do bitmap no disco.

Como compressão de bitmaps não será utilizado neste trabalho, apenas será mostrado um exemplo de cada tipo de compressão (4 e 8 bits) para ilustrar a diferença:

Compressão de 4 bits:

Compressed data	Expanded data
03 04	0 4 0
05 06	0 6 0 6 0
00 06 45 56 67 00	4 5 5 6 6 7
04 78	7 8 7 8
00 02 05 01	Move 5 right and 1 down
04 78	7 8 7 8
00 00	End of line
09 1E	1 E 1 E 1 E 1 E 1
00 01	End of RLE bitmap

Compressão de 8 bits:

Compressed data	Expanded data
03 04	04 04 04
05 06	06 06 06 06 06
00 03 45 56 67 00	45 56 67
02 78	78 78
00 02 05 01	Move 5 right and 1 down
02 78	78 78
00 00	End of line
09 1E	1E 1E 1E 1E 1E 1E 1E 1E 1E
00 01	End of RLE bitmap

Para terminar esse tópico que explica um pouco da teoria sobre bitmaps será apresentado um exemplo completo de um bitmap:

Win3DIBFile

BitmapFileHeader

Type 19778

Size 3118

Reserved1 0

Reserved2 0

OffsetBits 118

BitmapInfoHeader

Size 40

Width 80

Height 75

Planes 1

BitCount 4

Compression 0

SizeImage 3000

XPelsPerMeter 0

YPelsPerMeter 0

ColorsUsed 16

ColorsImportant 16

Win3ColorTable

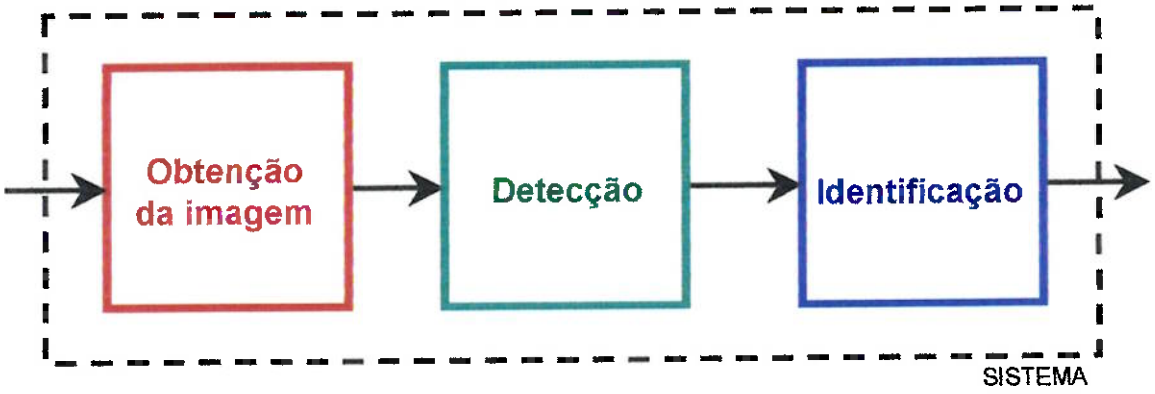
	Blue	Green	Red	Unused
[00000000]	84	252	84	0
[00000001]	252	252	84	0
[00000002]	84	84	252	0
[00000003]	252	84	252	0
[00000004]	84	252	252	0
[00000005]	252	252	252	0
[00000006]	0	0	0	0

[00000007]	168	0	0	0
[00000008]	0	168	0	0
[00000009]	168	168	0	0
[0000000A]	0	0	168	0
[0000000B]	168	0	168	0
[0000000C]	0	168	168	0
[0000000D]	168	168	168	0
[0000000E]	84	84	84	0
[0000000F]	252	84	84	0

DEFINIÇÃO DO SISTEMA

Definiremos as características básicas para o sistema que será desenvolvido. A necessidade a ser atendida é a de, automaticamente, detectar e reconhecer objetos estranhos a um ambiente que está sendo monitorado. Cabe ressaltar que definiremos objeto estranho como todo aquele objeto que em um instante anterior não pertencia ao ambiente. Sendo assim, poderíamos definir o sistema como uma caixa preta com as seguintes variáveis de entrada e saída.

Baseado no conceito de modulação (subdivisão do problema principal em subproblemas), e dadas as variáveis de entrada e saída acima definidas, o sistema será subdividido em três módulos básicos onde a saída de um módulo define a entrada do módulo seguinte. O primeiro módulo é o que recebe a imagem externa e faz uma adequação da imagem para a utilização no módulo seguinte. O segundo módulo procura uma alteração significativa na imagem presente da cena em relação a uma imagem tida como referência. Uma vez feita a detecção, passa-se ao terceiro módulo, onde o sistema busca identificar a forma anteriormente detectada. Esquematicamente, temos:

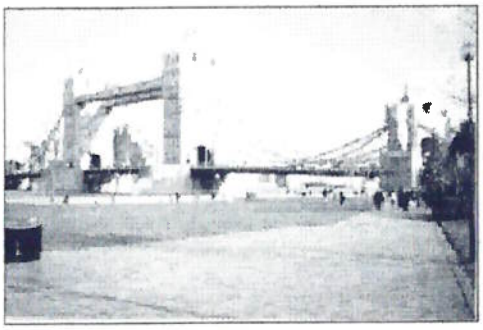


A obtenção de Imagens

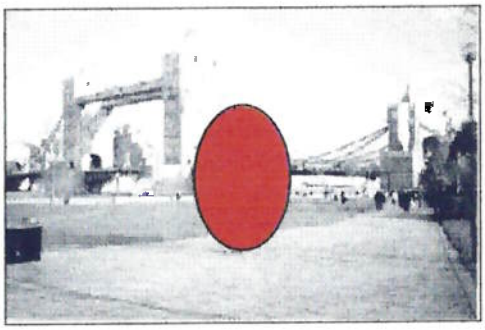
O sistema será implementado com uma câmera fixa, monitorando sempre o mesmo quadro. Assim, existirá apenas uma imagem de referência, que será atualizada quando o sistema for inicializado, para evitar que uma mudança de objetos estáticos (móveis, quadros, etc) provoque uma identificação errônea.

Apresentaremos duas fotos do mesmo local para ilustrar a obtenção das imagens.

Referência



Atual



No processo de obtenção e preparação da imagem para os módulos de detecção e identificação, um processo de "suavização" é efetuado nas imagens com dois objetivos básicos:

1) Homogeneizar a imagem, eliminando ruídos relativos a diferenças de iluminação no ambiente decorridas da inclusão de um novo objeto à cena.
2) Melhora as condições de contorno do objeto em estudo, em função dos algoritmos que serão usados para sua identificação

O processo de Detecção

Para efetuar a detecção de objetos novos à cena, decompomos a cor de cada pixel em suas componentes básicas, de modo a fazer uma comparação adequada. Em seguida, compara-se pixel a pixel e mede-se a diferença entre cada um dos componentes de cor. Uma vez encontrada uma diferença maior do que uma sensibilidade pré-definida, o sistema assume que encontrou um elemento que não pertencia à cena original.

O processo de Identificação

Para o processo de identificação das formas das figuras serão levadas em consideração apenas características geométricas das mesmas em detrimento a uma primeira solução encontrada para o problema, essa solução não adotada

previa a utilização de um banco de dados de formas geométricas para serem comparadas com a forma detectada na cena. A não adoção deste método se deu devido a dificuldade no ajuste de escala entre as figuras e a orientação das mesmas em relação aos eixos de referência.

Este módulo está estruturado de modo a primeiramente tentar encontrar um círculo e, no caso de insucesso, tenta identificar polígonos, até um máximo de seis lados (hexágono), sendo que isso foi definido dessa maneira, pois um polígono (regular) com mais de seis lados poderia ser aproximado como um círculo.

Uma vez definida a região de controle, a sensibilidade de comparação e respeitando-se as premissas adotadas no início do projeto, o processo de identificação das formas se dá como descrito abaixo nos diagramas NS.

DIAGRAMAS NS

Neste tópico serão explicados os seguintes algoritmos:

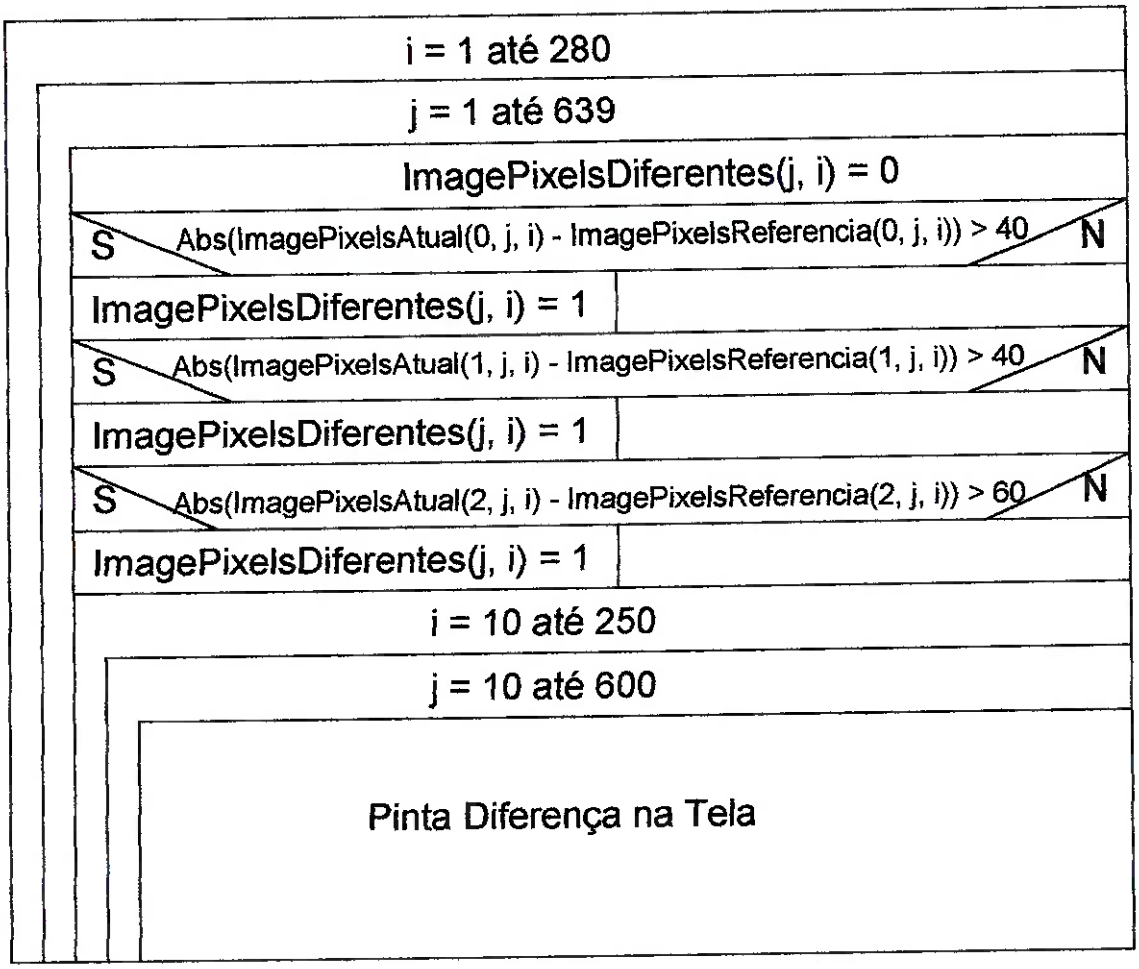
- Suavização de imagens;
- Image Difference;
- Cálculo do baricentro;
- Reconhecimento de Formas Circulares;
- Reconhecimento de Formas não Circulares;
- Achavértices;
- Limpaentredois;
- Verifica_vizinhos;

Suavização de Imagens

Inicia as variáveis
Percorre a figura de referência e atual
Para cada cor de pixel faz se uma média entre o pixel(i,j) e seus vizinhos atribuindo o valor encontrado às variáveis vermelho, verde e azul.
As variáveis red, green e blue são declaradas
red = pixel& Mod 256 green = ((pixel& And &HFF00) / 256&) Mod 256& *blue* = (pixel& And &HFF0000) / 65536
ImagePixelsAtual(0, j, i) = red ImagePixelsAtual(1, j, i) = green ImagePixelsAtual(2, j, i) = blue

Image Difference

Este algoritmo é responsável pela subtração entre as imagens. Essa subtração é feita em cada cor dos pixels, ou seja, os pixels são decompostos nas suas cores básicas (vermelho, azul, verde) e são subtraídas as respectivas cores pixel a pixel.



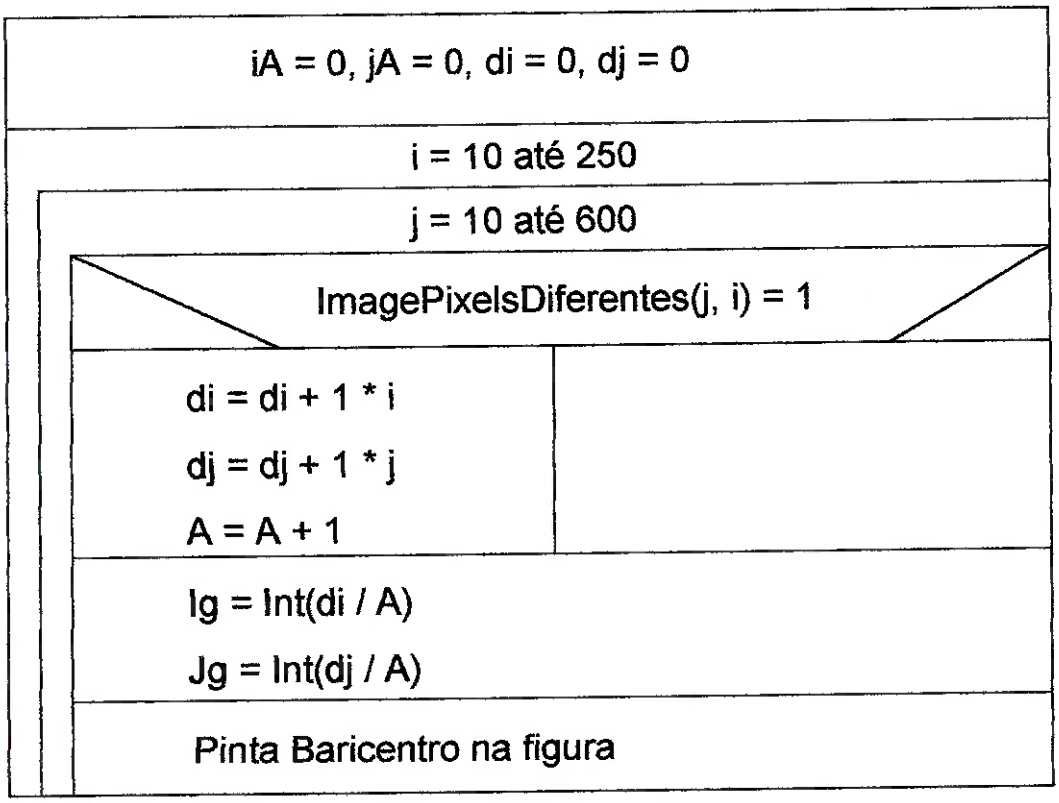
Este algoritmo percorre uma matriz tridimensional onde cada dimensão corresponde a uma cor. Percorrendo a matriz, o algoritmo compara a diferença entre os respectivos pixels com um valor de sensibilidade definido. Caso a

diferença seja maior que essa sensibilidade então uma matriz bidimensional binária é montada de maneira a identificar os pixels diferentes como 0 ou 1. Esta diferença pode estar em qualquer uma das cores, a matriz binária resultante terá a forma do objeto a ser identificado.

Essa matriz binária é então desenhada na tela de maneira a possibilitar o reconhecimento da figura.

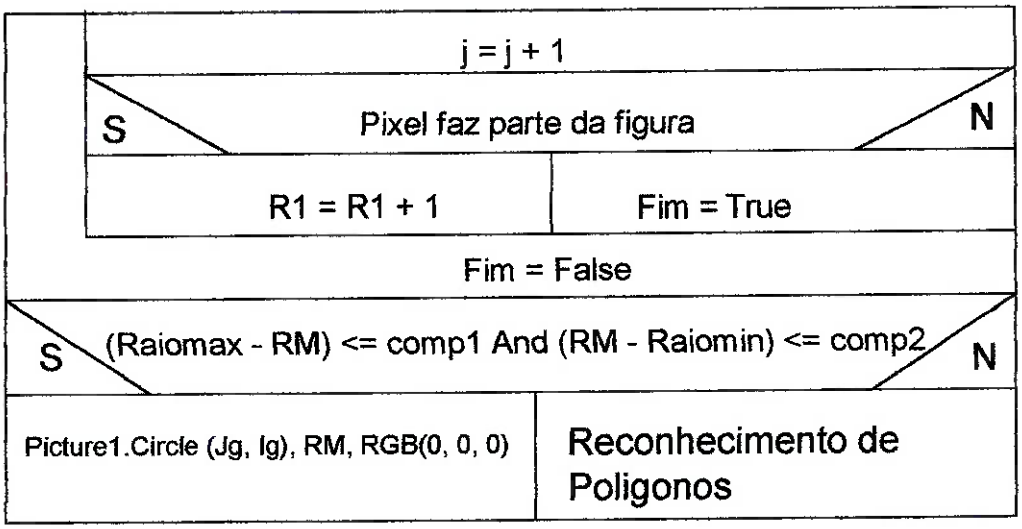
Cálculo do Baricentro

Para o cálculo do baricentro é utilizado a matriz binária anterior.



A matriz é percorrida de maneira a permitir o armazenamento do número de pixels nas variáveis di e dj correspondentes as coordenadas i e j respectivamente. A variável A armazena o total de pixels percorridos de maneira a permitir o cálculo do baricentro obtendo Ig e Jg. Este algoritmo também pinta o baricentro na tela.

Reconhecimento de formas circulares



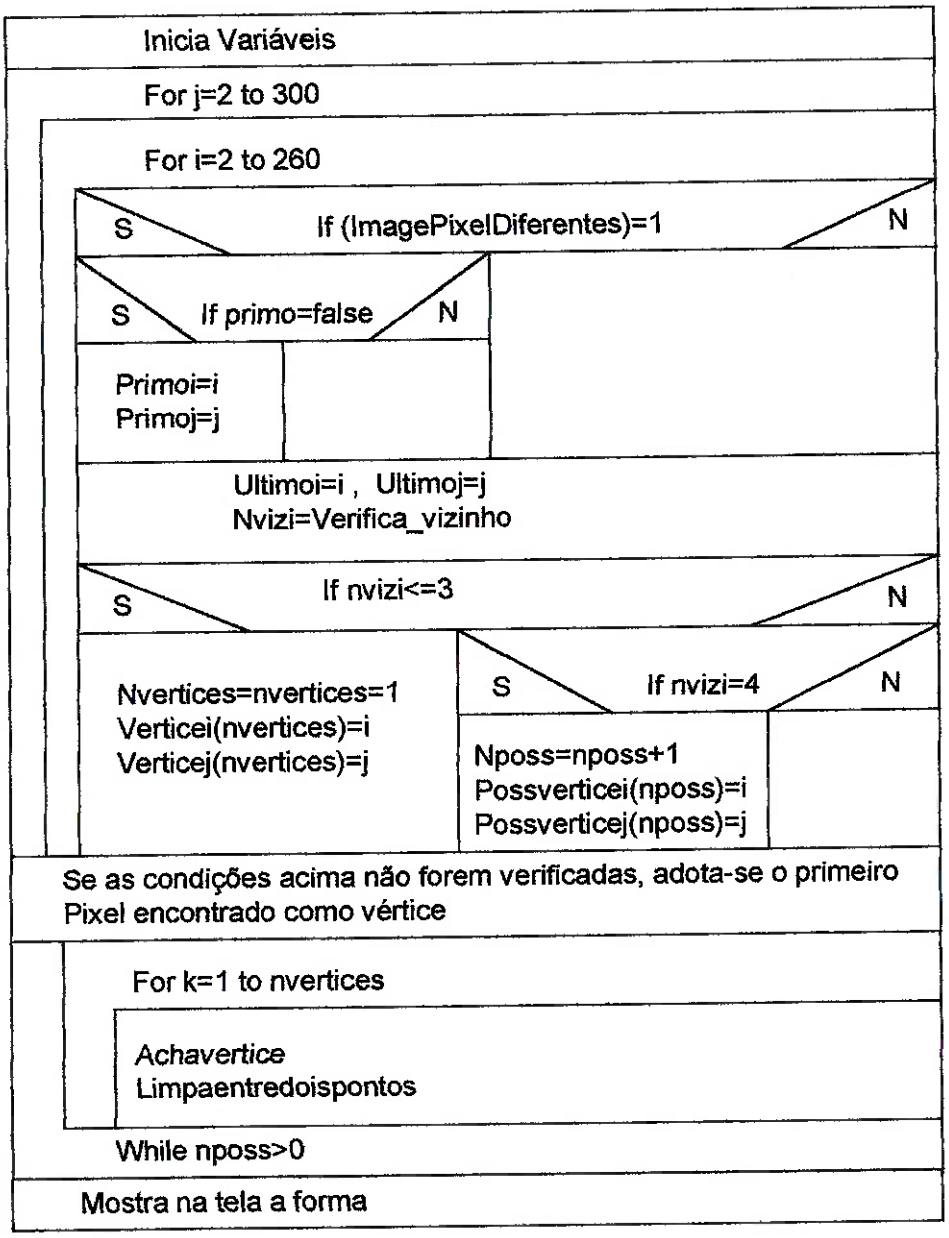
Este algoritmo percorre a matriz binária identificando os pixels como pertencentes ou não a figura montando oito raios em orientações diferentes.

Com esses oito raios é determinado um raio médio que será comparado com os rios máximo e mínimo desta figura. Definido os parâmetros de comparação, o seguinte teste é feito:

$(RaioMáximo - RaioMédio) \leq 5$ e $(RaioMédio - RaioMínimo) \leq 5$

Caso esta verificação seja verdadeira então a figura é reconhecida como um círculo e desenhada na tela com seu baricentro e raio médio, caso contrário começa a verificação de polígonos.

Reconhecimento de Polígonos

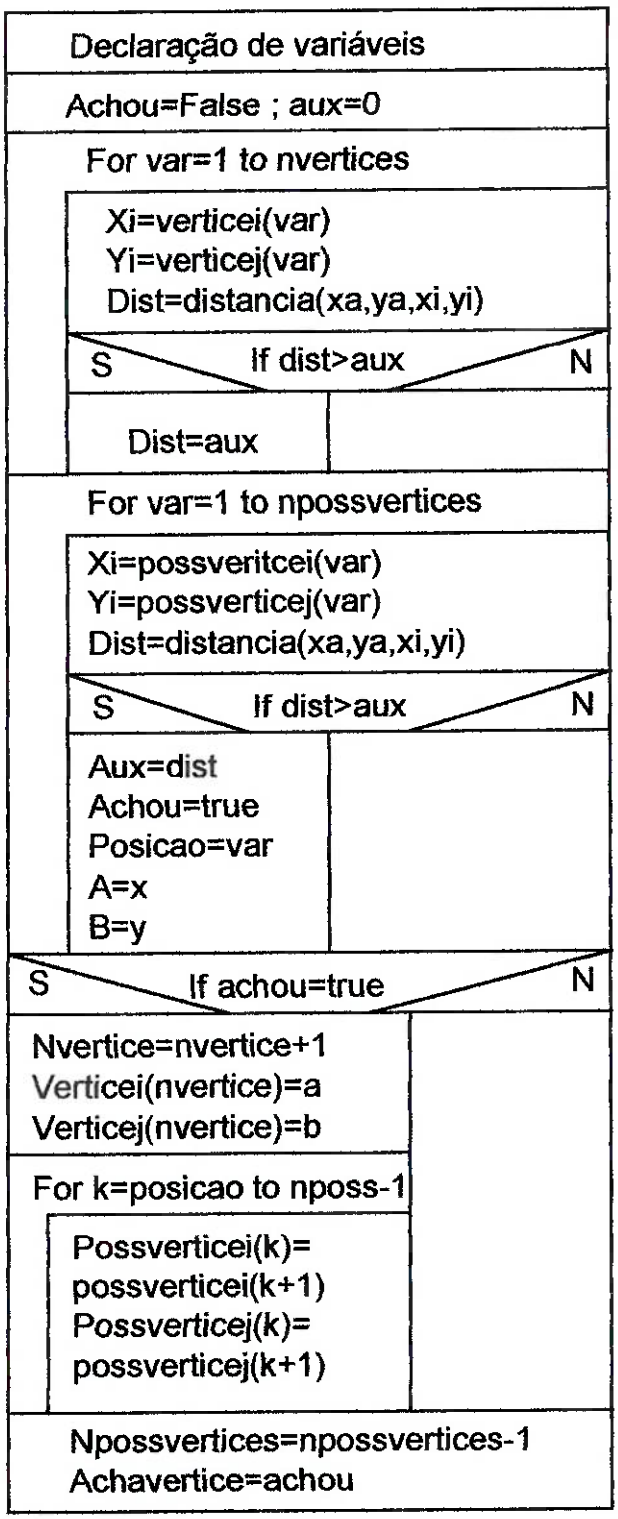


Este algoritmo percorre a matriz binária, caso encontre um pixel armazena como sendo o primeiro e este primeiro pixels será considerado um vértice caso não haja pixels com número de vizinhos igual a três. Este pixel é armazenado nas variáveis primo1 e primoj, o ultimo pixel encontrado é armazenado em ultimoi e ultimoj. Encontrado um pixel com número de vizinhos igual a três o número de vértices certos é incrementado e sua coordenada é armazenada em dois vetores (verticesi(n) e verticesj(n)). Se o número de vizinhos for igual a quatro, então este pixel é tratado com um possível vértice e sua coordenada é armazenada em outros dois vetores (possverticei(n) e possverticej(n).

Caso nenhum pixel possa ser considerada vértice certo (nvizi <= 3) então o algoritmo considera o primeiro pixel como vértice.

Este algoritmo chama uma função Achavertice que irá decidir se os pixels com quatro vizinhos serão ou não vértices e os que não forem são eliminados por outra função chamada Limpaentredoispontos.

Achavertice

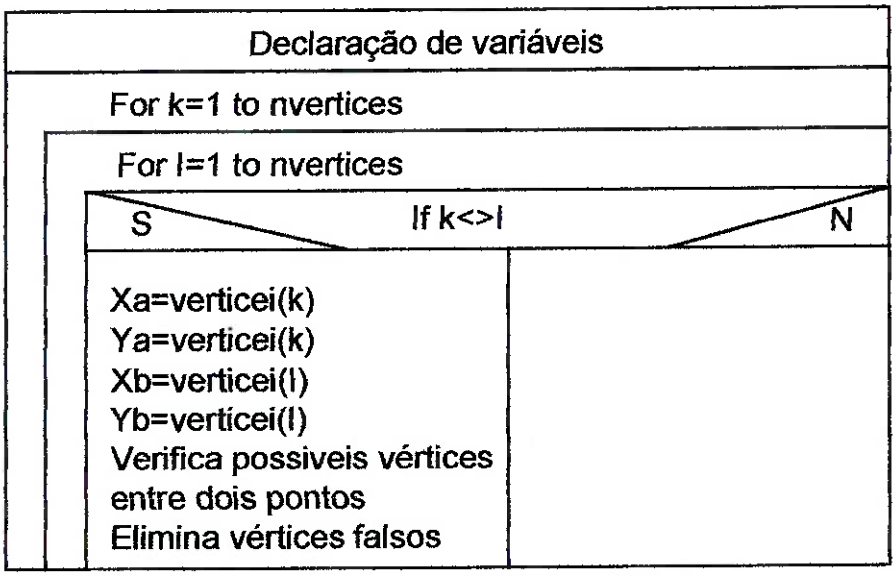


A função achavertice é responsável por determinar quais dos possíveis vértice são vértices reais. Os possíveis vértices são os pixels que possuem quatro vizinhos com valor 1 na matriz binária.

Esta função além de determinar quais são os vértices também guarda em vetores apropriados as respectivas coordenadas desse vértices e decrementa o número de possíveis vértices.

Esta função retorna um valor booleano indicando se encontrou um vértice verdadeiro ou não ao analisar determinado pixel. O algoritmo funciona como descrito no diagrama NS acima.

Limpaentredoispontos



A função limpaentredoispontos é responsável pela eliminação de pixels que forma considerados como possíveis vértices.

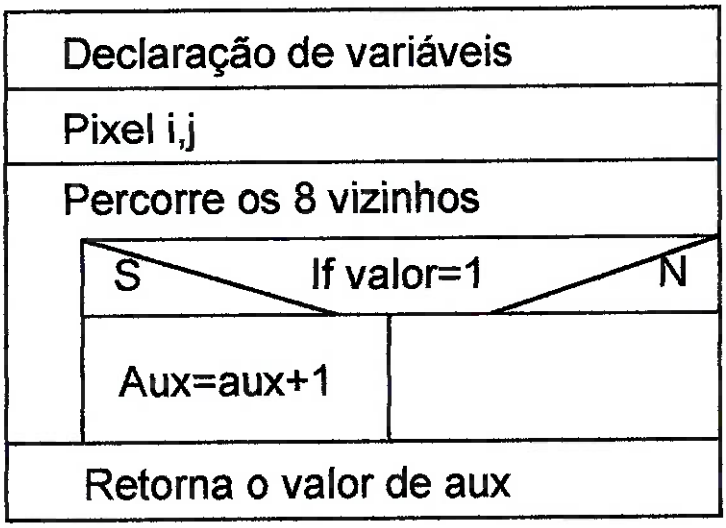
A função faz uma verificação dos vértices reais e entre dois deles ela percorre uma faixa eliminando candidatos a vértice (pixels com quatro vizinhos com valor 1).

Desta forma são apenas contabilizados vértices verdadeiros para o reconhecimento da forma e determinação de suas características geométricas.

Essa função deve ser aplicada a todos os vértices reais eliminando assim qualquer chance de erro de contagem. O Algoritmo trabalha de acordo com o diagrama acima.

Uma função que auxilia no processo é a Verifica_vizinhos que será explicada no próximo diagrama NS.

Verifica_Vizinhos



Esta função percorre a vizinhança de um pixel fornecido como parâmetro verificando a existência ou não de pixels com valor 1 na matriz binária.

Em existindo pixels nessas condições incrementa-se uma variável auxiliar que retornará este número.

Este numero é utilizado em boa parte dos algoritmos neste software e é de extrema importância na determinação da forma e na definição dos parâmetros de sensibilidade para polígonos.

A determinação dessa sensibilidade será mostrada no próximo tópico

DETERMINAÇÃO DA SENSIBILIDADE

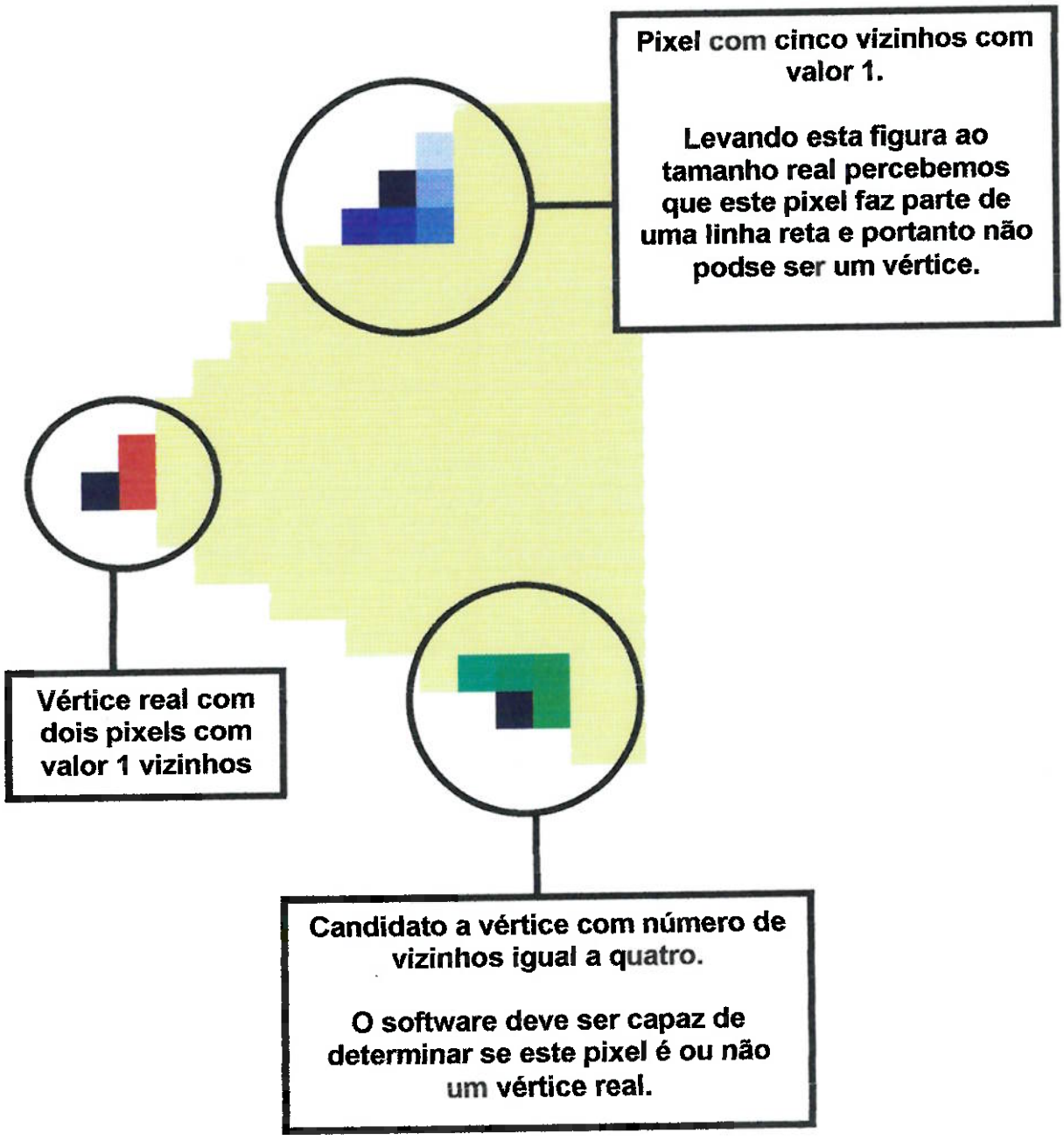
A sensibilidade é determinada pelo número de pixels vizinhos para o caso de polígonos ou pelo número de pixels a mais ou a menos na relação entre rios máximo e mínimo com o raio médio.

Consideramos como vértice todos os pixels que possuem três ou menos pixels vizinhos com valor igual a um na matriz binária após a suavização.

Os possíveis vértices possuem quatro vizinhos com esse valor e os pixels com cinco ou mais vizinhos não são considerados como possíveis vértices ou porque são internos a figura (todos possuem valor 1) ou por serem sempre intermediários.

O exemplo abaixo é uma ampliação da forma reconhecida como triângulo na demonstração de resultados do software. Nele podemos reparar que o vértice real nesse caso possui apenas dois vizinhos com valor 1, os pixels que são considerados como possíveis vértices possuem valor igual a quatro e o pixel com número de vizinhos igual a cinco faz parte de uma linha reta, ou seja, não pode nem ao menos ser um possível vértice.

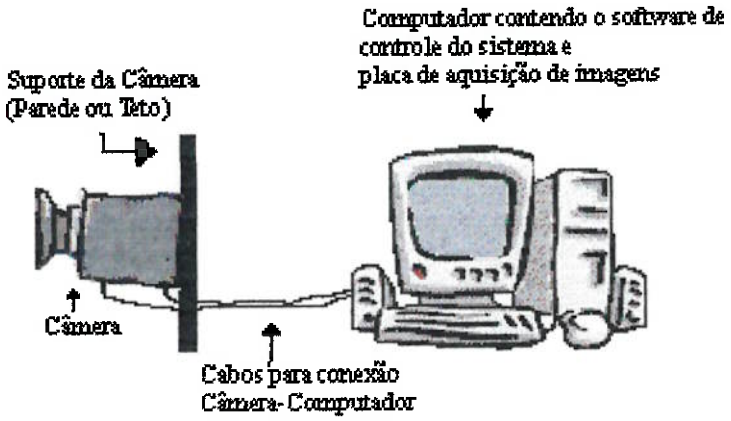
Vejamos:



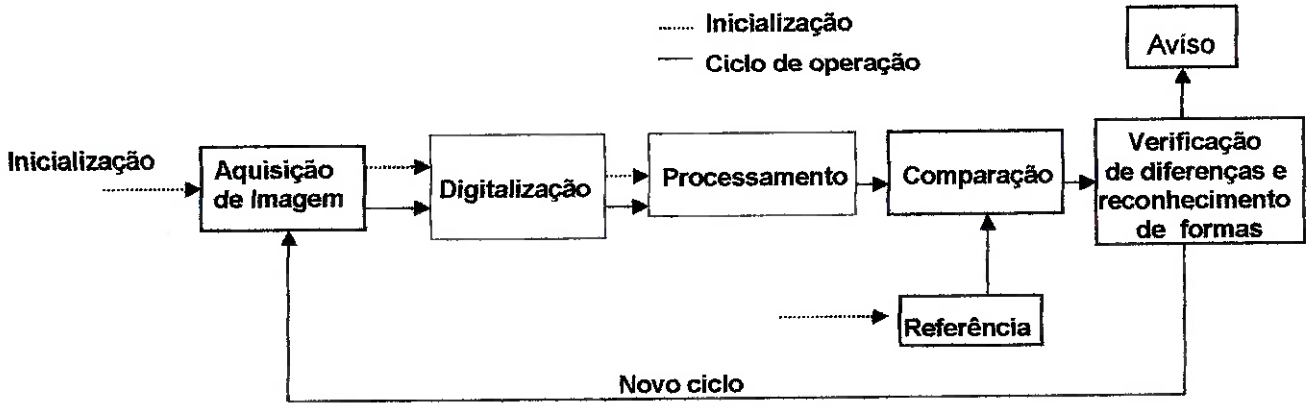
Lembrando apenas que esta figura é uma ampliação do canto do triângulo reconhecido com um zoom de 1600% para que se possa visualizar os pixels e su comportamento.

REPRESENTAÇÃO DO SISTEMA COMPLETO

O sistema será basicamente composto de uma câmera conectada a um computador, placa de aquisição de imagens e um software que controla o processo de aquisição, análise e armazenamento de imagens e disparo de alarme. Este sistema está representado abaixo.



O sistema pode ser modelado segundo o diagrama de blocos mostrado na figura a seguir:



Este diagrama visa fornecer informações sobre o fluxo de informações no sistema.

Sistemas de aquisição

São sistemas físicos destinados a produzir (ou captar) imagens. Nos sistemas de aquisição, a imagem é formada num plano através de lentes. Nesse plano são colocados sensores que medem a intensidade da luz incidente. Neste sistema é composto pela câmera.

Digitalização

A digitalização das imagens será feita por uma placa comercial, motivo pelo qual não trataremos de teoria e métodos de discretização

Processamento das Imagens

Por processamento digital de imagens entende-se a análise e a manipulação de imagem através do computador, tendo como finalidade extrair informações da imagem (dessemelhança) ou transformá-la.

Comparação

A comparação entre as duas imagens (atual e referência) será feita através da Diferença de Imagens (Image Diference). Esse processo tanto pode ser efetuado pixel por pixel como também por amostragem (por exemplo, linha sim, linha não, coluna sim, coluna não), o que representa um processamento mais rápido. Desse processo resultará uma medida de dessemelhança, cujas informações serão utilizadas para reconhecimento de formas padronizadas.

Armazenamento

As imagens, quando digitalizadas, necessitam de um número elevado de bits de memória para armazená-las, diretamente influenciado pela resolução da digitalização e também pelo formato em a imagem é armazenada . O banco de dados utilizado para o reconhecimento de padrões (formas) guardará figuras mono-cromáticas, sendo representado por matrizes binárias, que podem ser armazenadas na forma de quadtree's, para economia de memória.

Hardware

A parte de hardware deste sistema em que foram realizados os testes consiste em um computador Pentium 200 Mhz, 32 RAM rodando windows NT 4 Workstation

como Sistema Operacional, uma Câmera de Vídeo Monocromática fixa, uma Placa de Aquisição de imagens da Data Translation, Inc, modelo DT3153 (maiores detalhes sobre a placa de aquisição no Anexo C).

Na configuração utilizada para a realização dos testes obtivemos imagens no formato Bitimap com 640 pixels de largura por 480 pixels de altura, com uma definição de 189 pixels/cm.

Software

Para a construção do software de tratamento das imagens foi utilizado a linguagem Basic e compilador orientado a objetos Visual Basic 6.0 da Microsoft.

O SOFTWARE

O software desenvolvido tem interface gráfica facilitando o seu manuseio, para isso foi desenvolvido em módulos utilizando os princípios de programação orientada a objetos.

O objetivo deste software é capturar imagens de um determinado ambiente como referência para poder identificar variações nesta imagem.

Estas variações podem ser simples alterações na iluminação local bem como a presença de um objeto estranho.

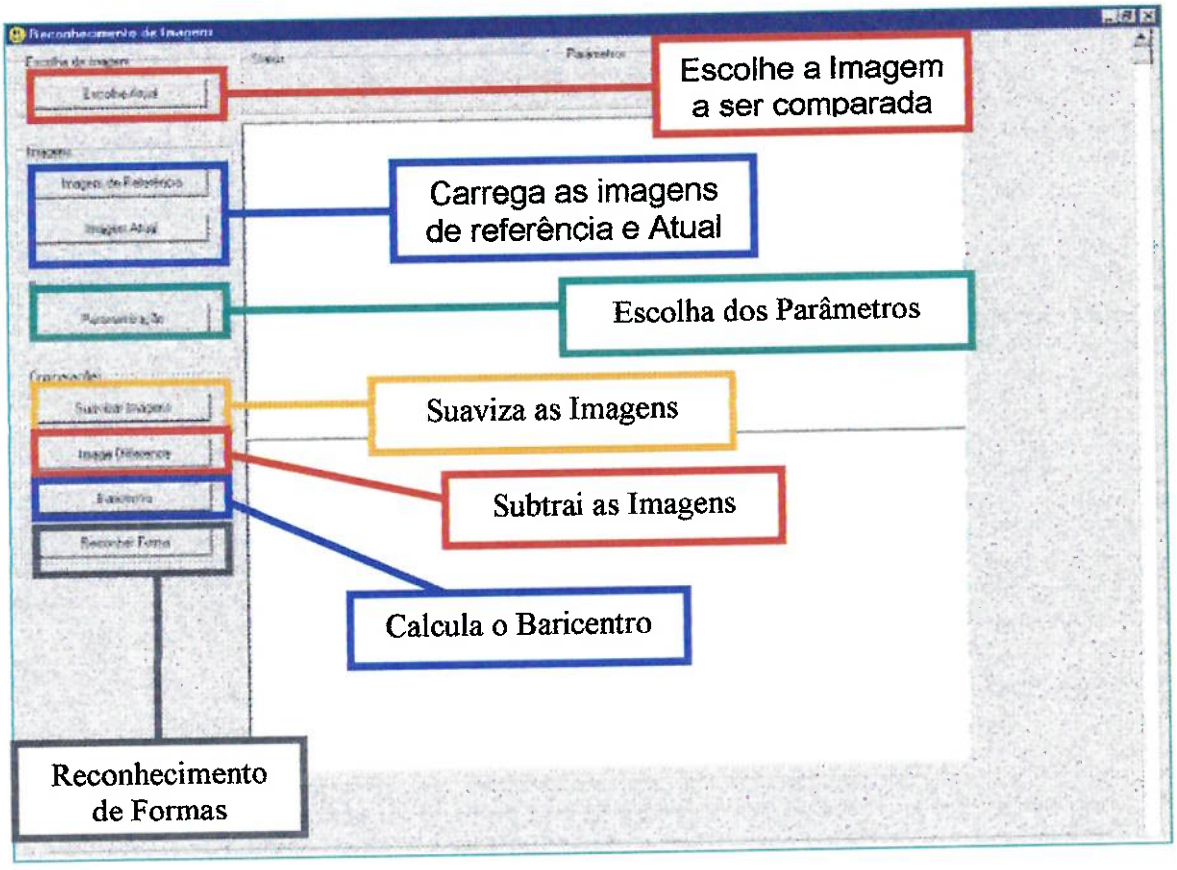
A capacidade de reconhecimento deste software se limita a formas geométricas simples tais como círculos, triângulos, quadriláteros, pentágonos, hexágonos e assim por diante. Essas formas não necessitam necessariamente serem regulares.

Para o desenvolvimento dos algoritmos foram estabelecidas algumas premissas listadas abaixo:

- Todas as figures devem ser não côncavas;
- Não podem ser furadas;
- No quadro não pode haver mais do que uma figura;

- As cores do objeto não podem influenciar na identificação e reconhecimento da forma;
- Variações na iluminação local não pode influenciar na identificação e reconhecimento da forma;

A interface com o usuário é a mostrada abaixo:



Interface Gráfica

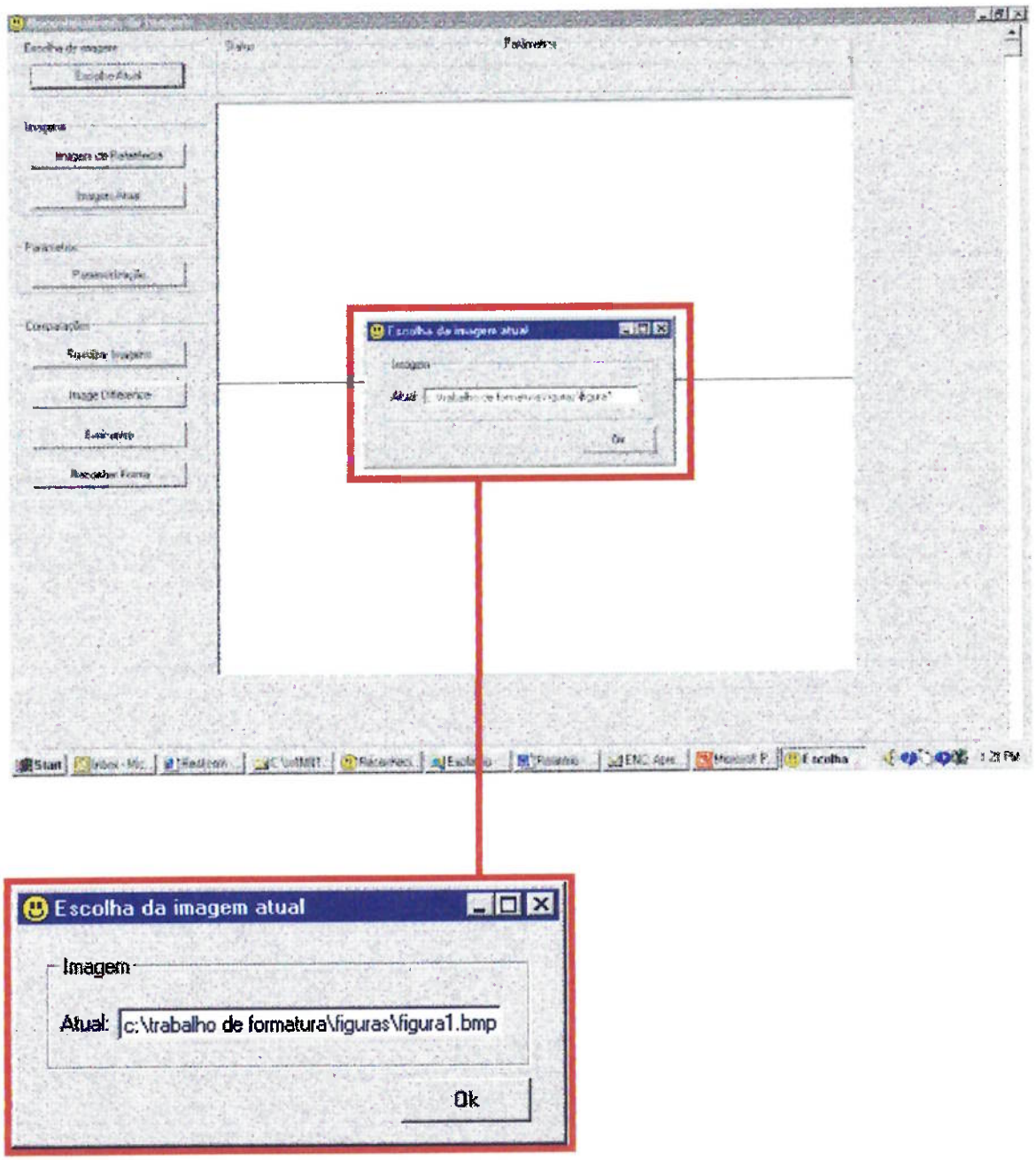
Este software utiliza algoritmos baseados em características geométricas dos objetos a serem reconhecidos.

São elas:

- Baricentro;
- Vértices e suas coordenadas;
- Distâncias entre coordenadas;
- Nos círculos utilizamos os raios;

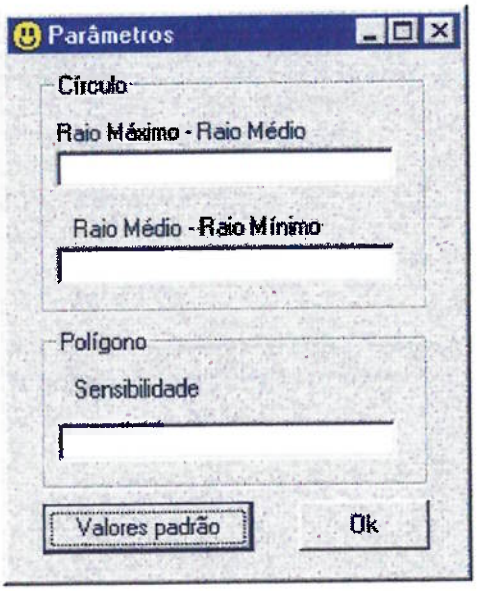
Mais detalhes de como isso é feito será apresentado no tópico que explica os algoritmos, que foi apresentados na forma de diagramas NS.

No software para efeito de teste pode-se escolher o arquivo a ser analisado conforme a figura abaixo:



No software também é possível *setar* parâmetros que definem a precisão do reconhecimento das formas.

Isto é feito de acordo com a seguinte interface:



Onde é possível determinar parâmetros para círculos e polígonos ou apenas adotar os valores padrão. Esses padrões são:

- Raio Máximo - Raio Médio = 5
- Raio Médio - Raio Mínimo = 5
- Sensibilidade = 3

Esses valores foram determinados de forma de forma empírica através de iterações em diversos casos.

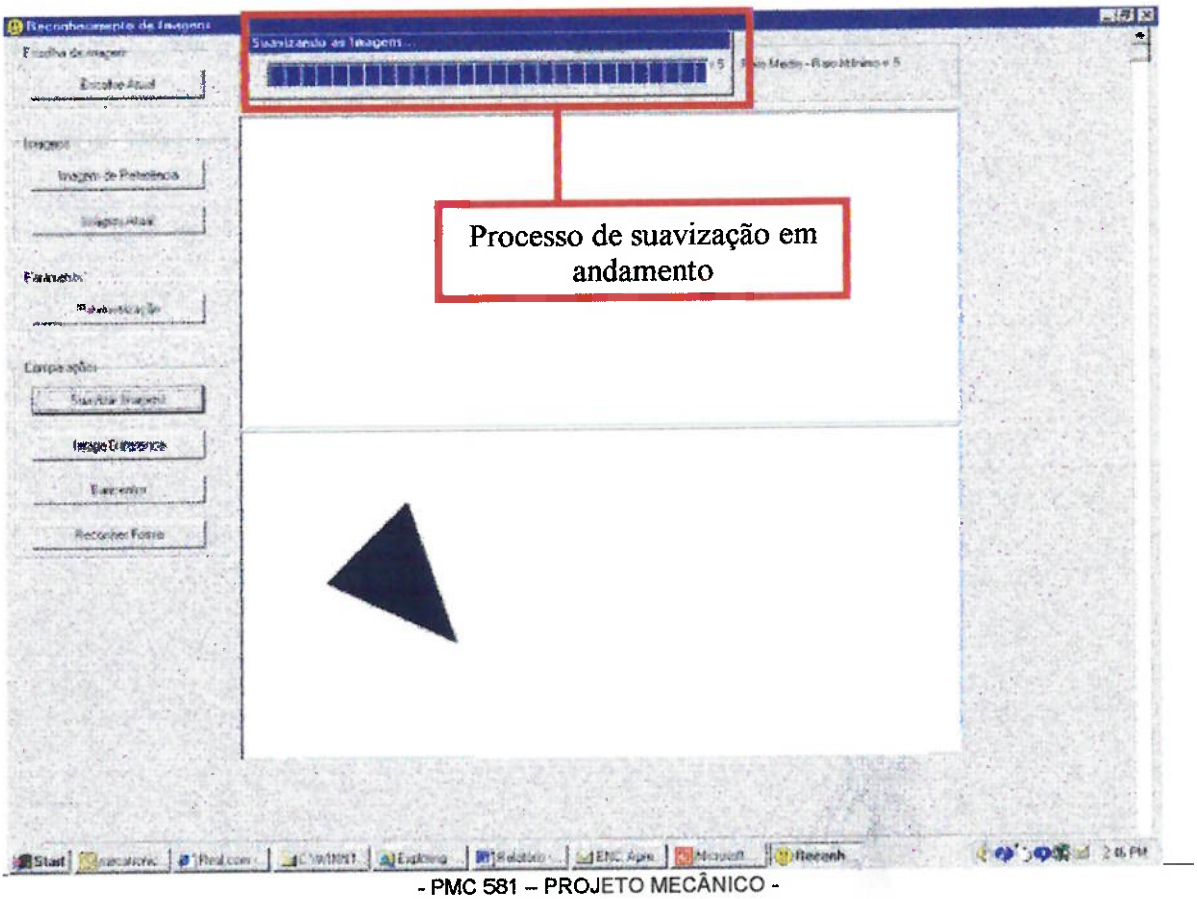
Para exemplificar o funcionamento do software foi escolhido algumas figuras que serão simuladas em algumas etapas do processo de reconhecimento.

Essas figuras são triângulo, quadrilátero, pentágono, hexágono e círculo.

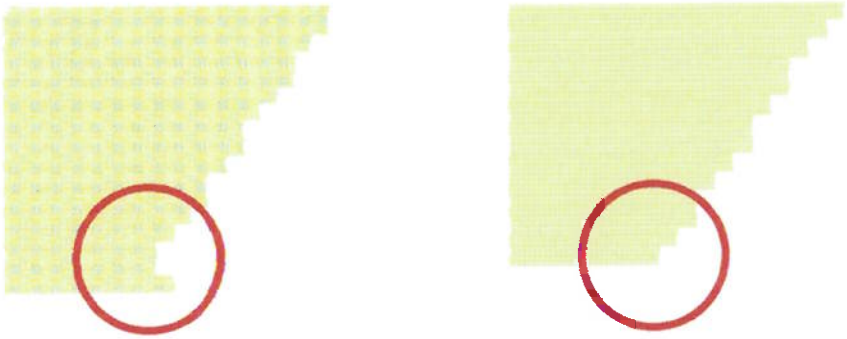
Reconhecimento:

Após a escolha da figura e carregá-las no software é necessário um processo de tratamento de suavização de imagens, este processo corresponde a eliminação de irregularidades ao nível de pixels.

A suavização se dá na interface mostrada abaixo:

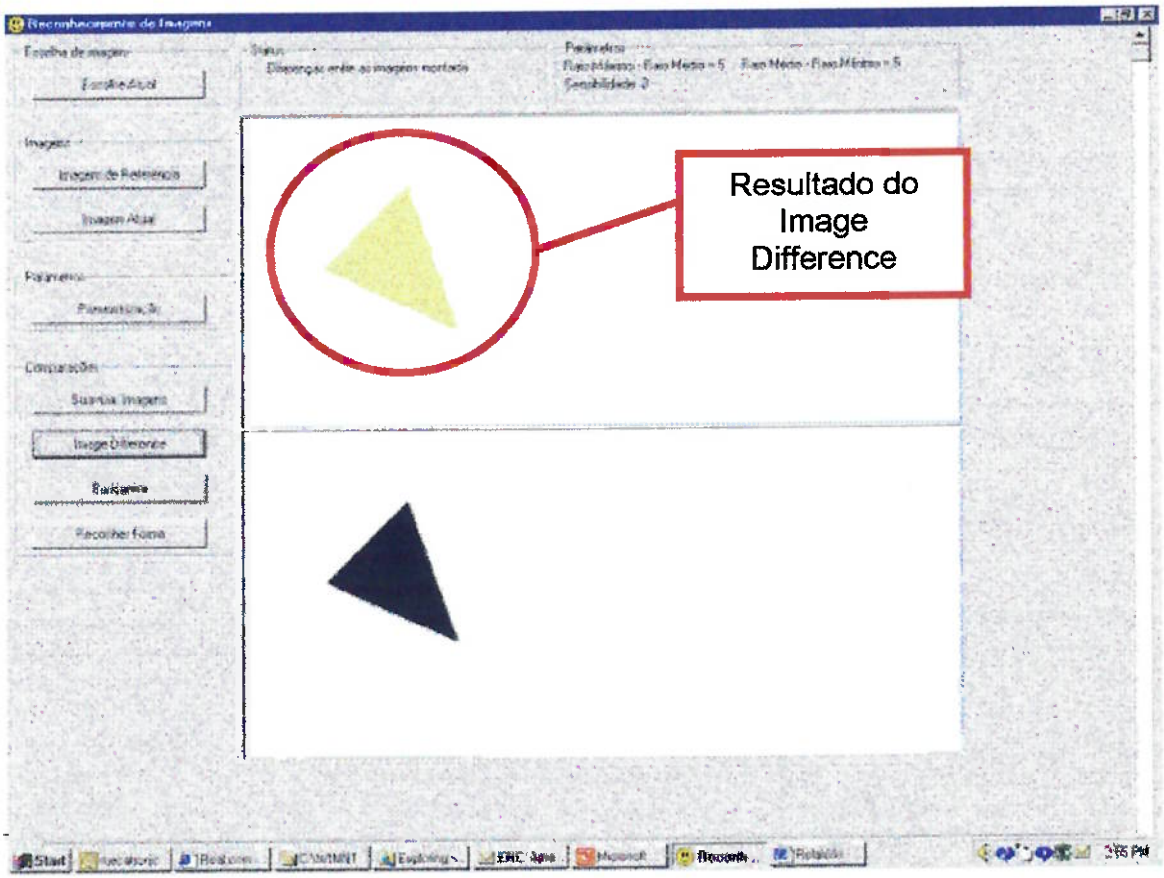


Em detalhe está a diferença nos pixels antes e depois da suavização:



Uma vez feita a suavização segue o processo de Image Difference na qual a imagem de referência e a que está sendo comparada são subtraídas.

Na interface:

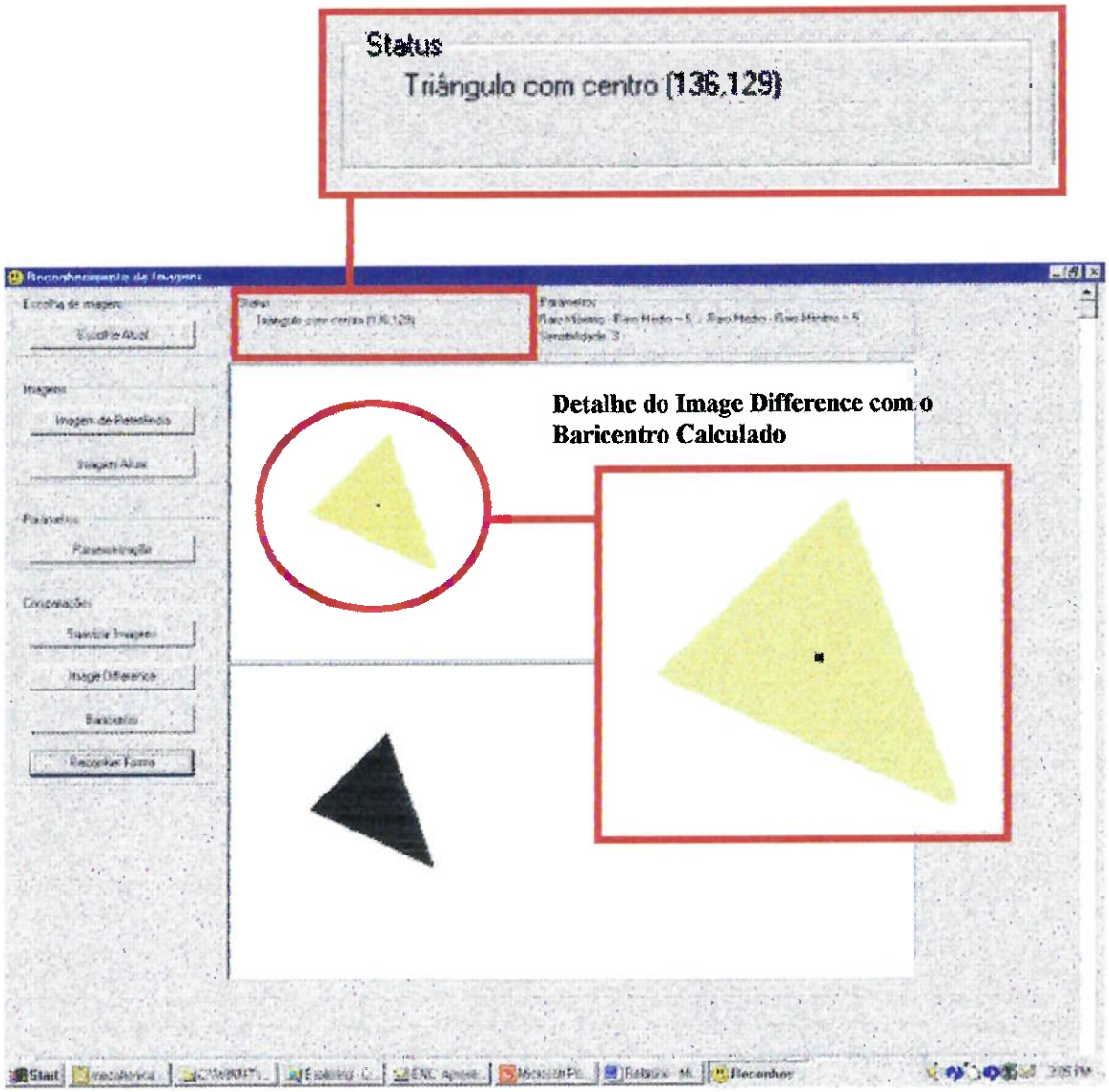


Onde o triângulo em cor amarela representa o resultado da subtração.

O baricentro é calculado e desenhado na região que representa a diferença entre os quadros.

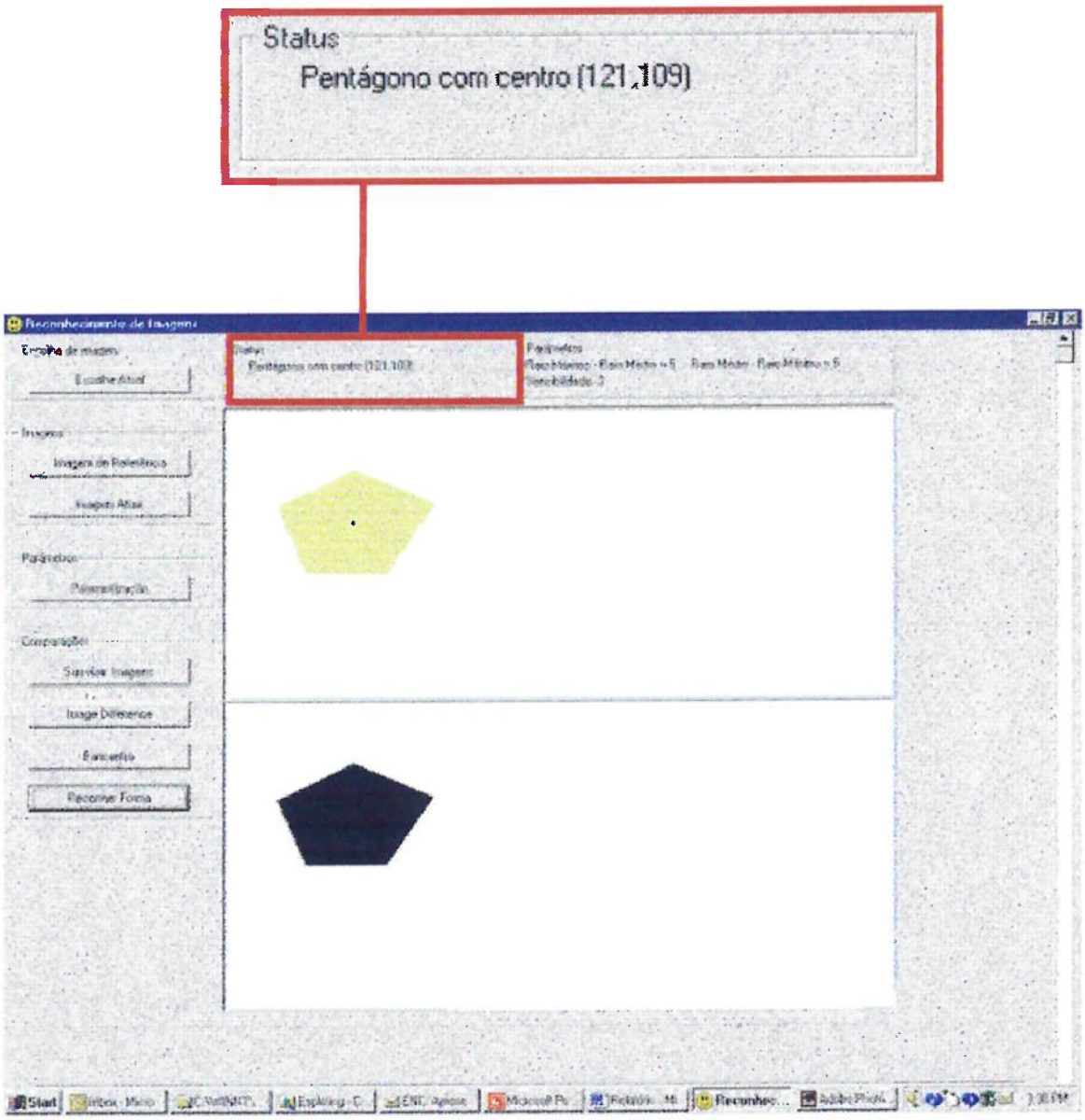
Os algoritmos que responsáveis pela suavização e pelo cálculo do baricentro serão explicados no tópico que trata dos diagramas NS.

Abaixo está a interface mostrando o baricentro desenhado bem como o resultado do processo de reconhecimento.

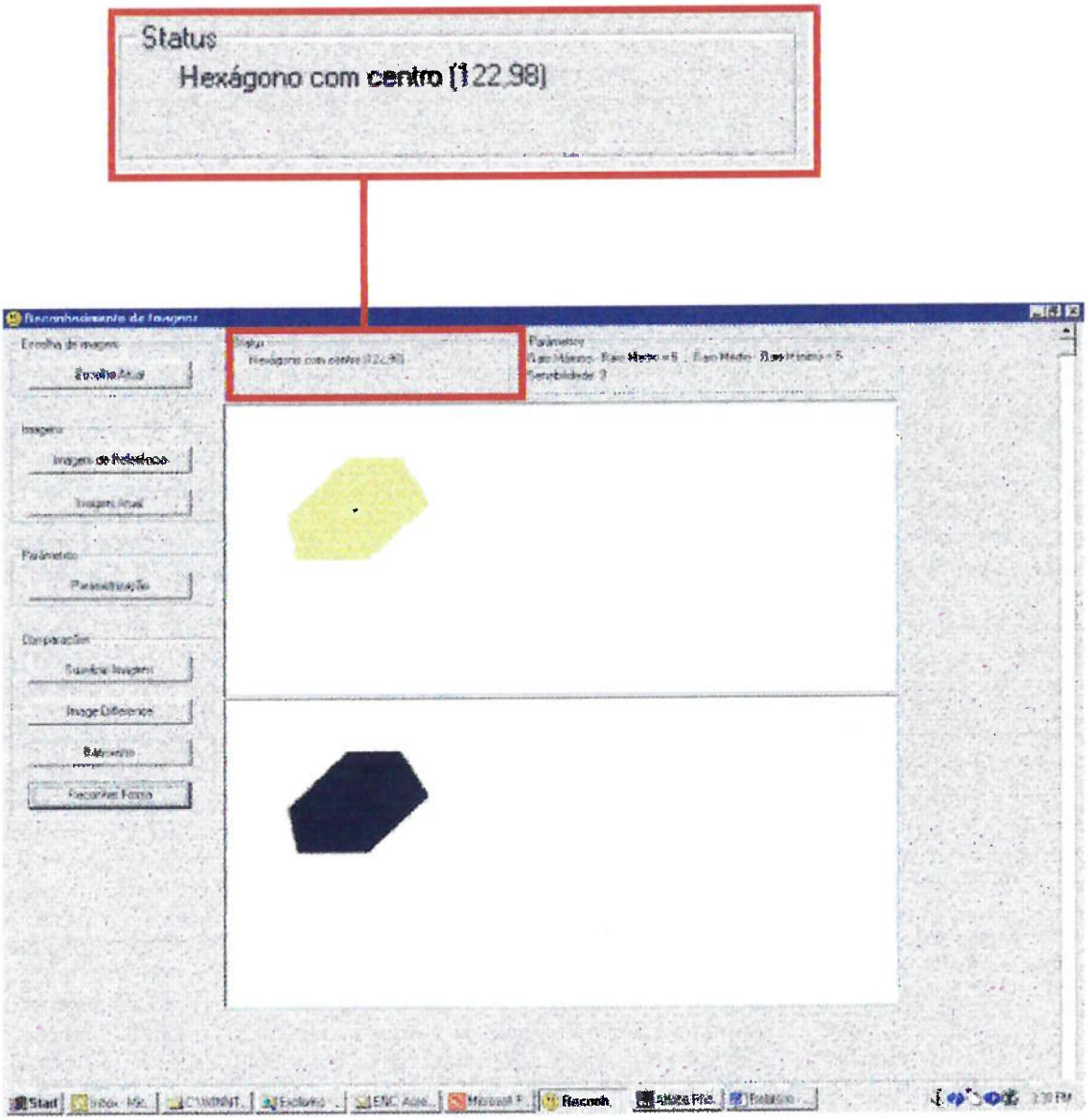


Para ilustrar melhor o funcionamento do software será mostrado a interface final de mais alguns objetos.

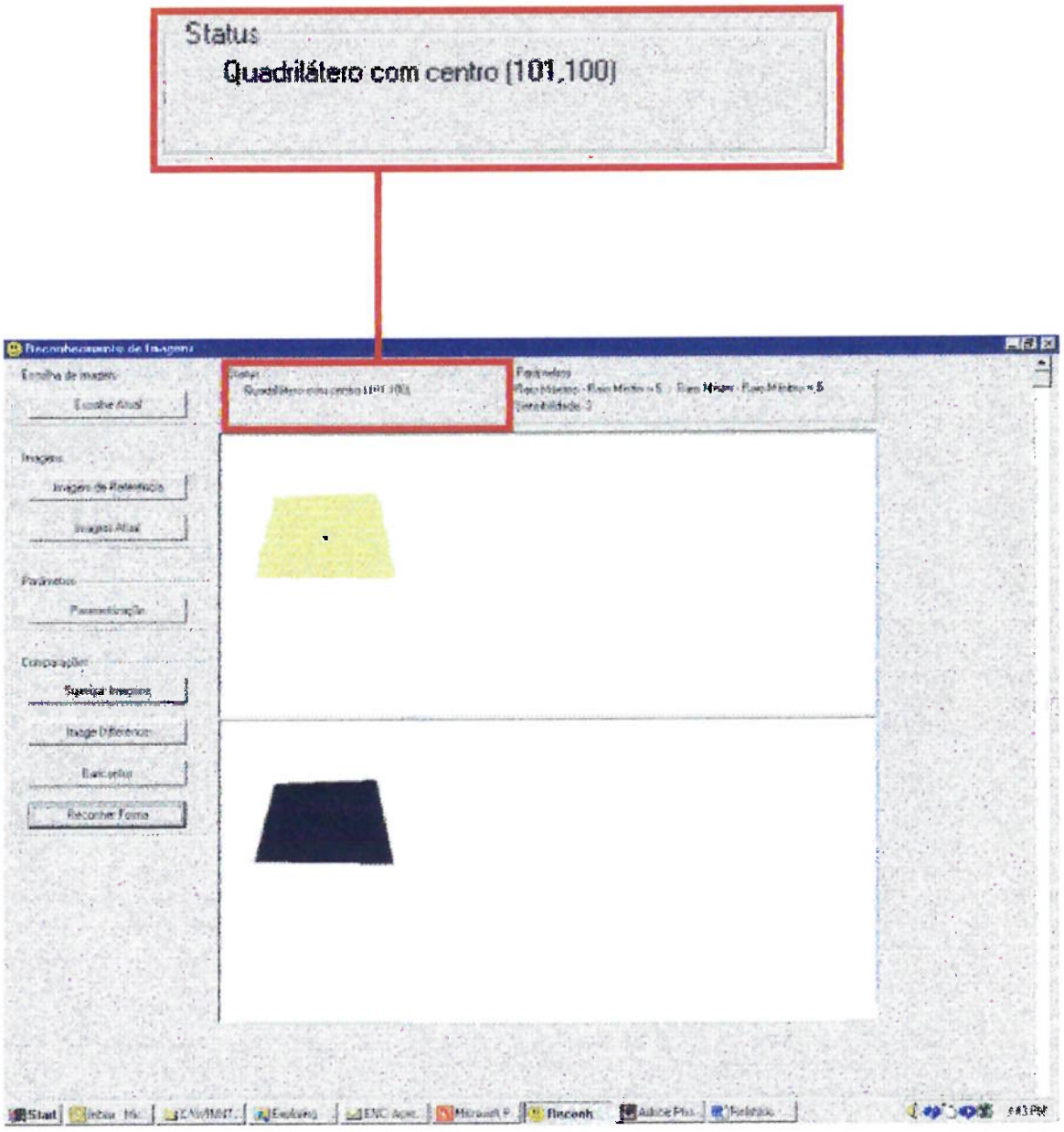
Para um pentágono qualquer teremos como saída sua identificação e determinação do seu baricentro, isso vale para as demais figuras:



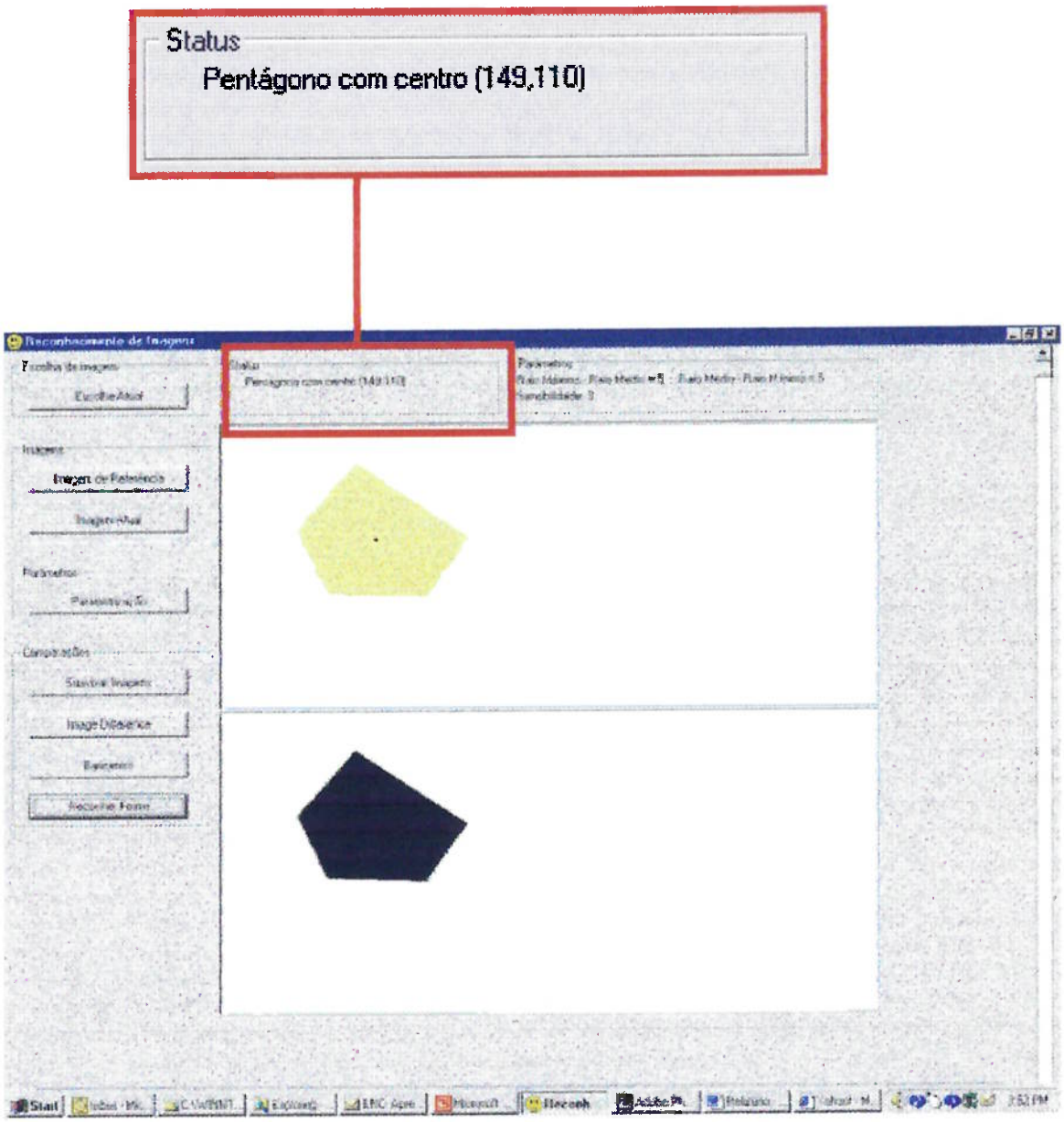
Para um hexágono qualquer:



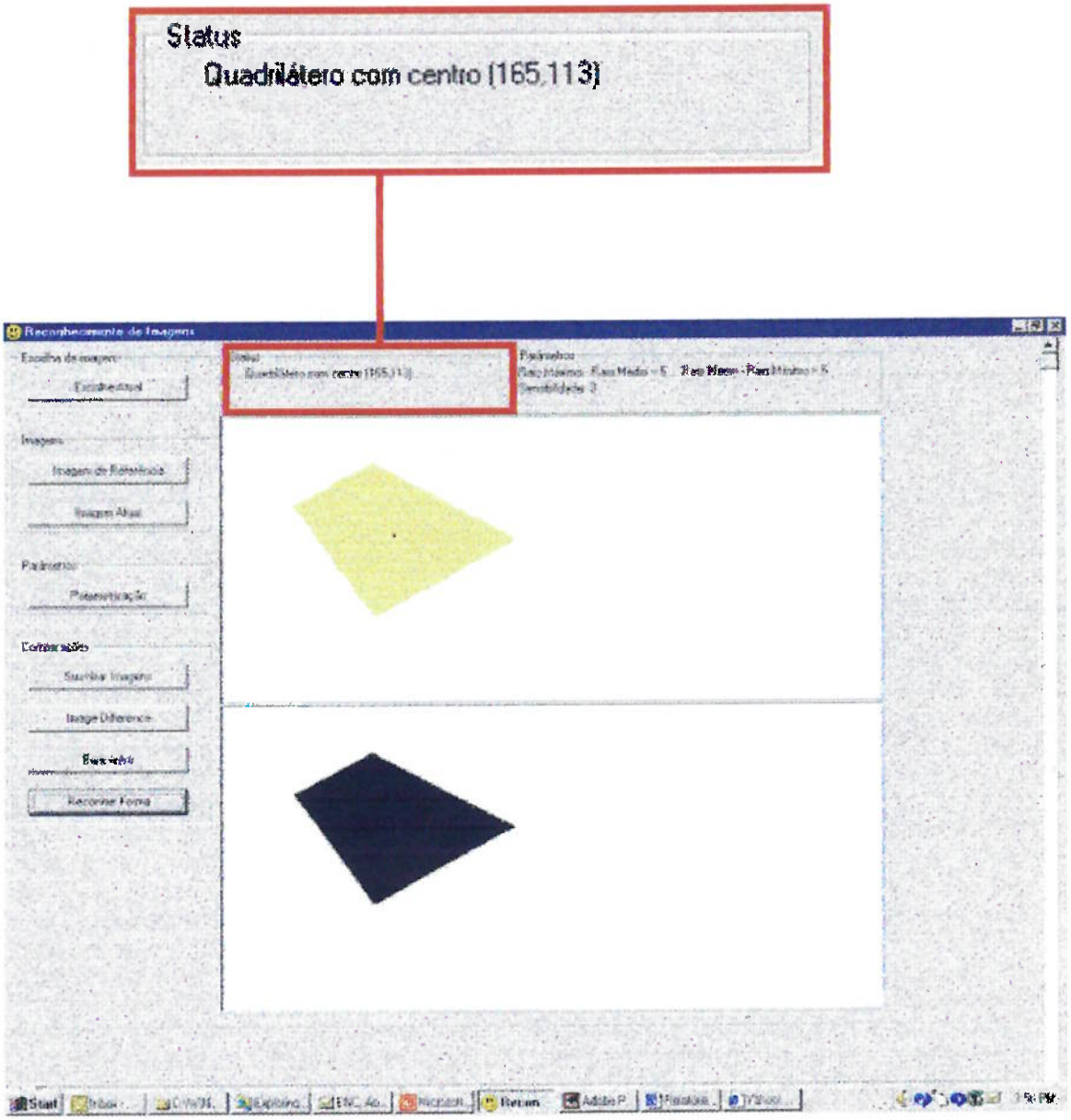
Para um quadrilátero:



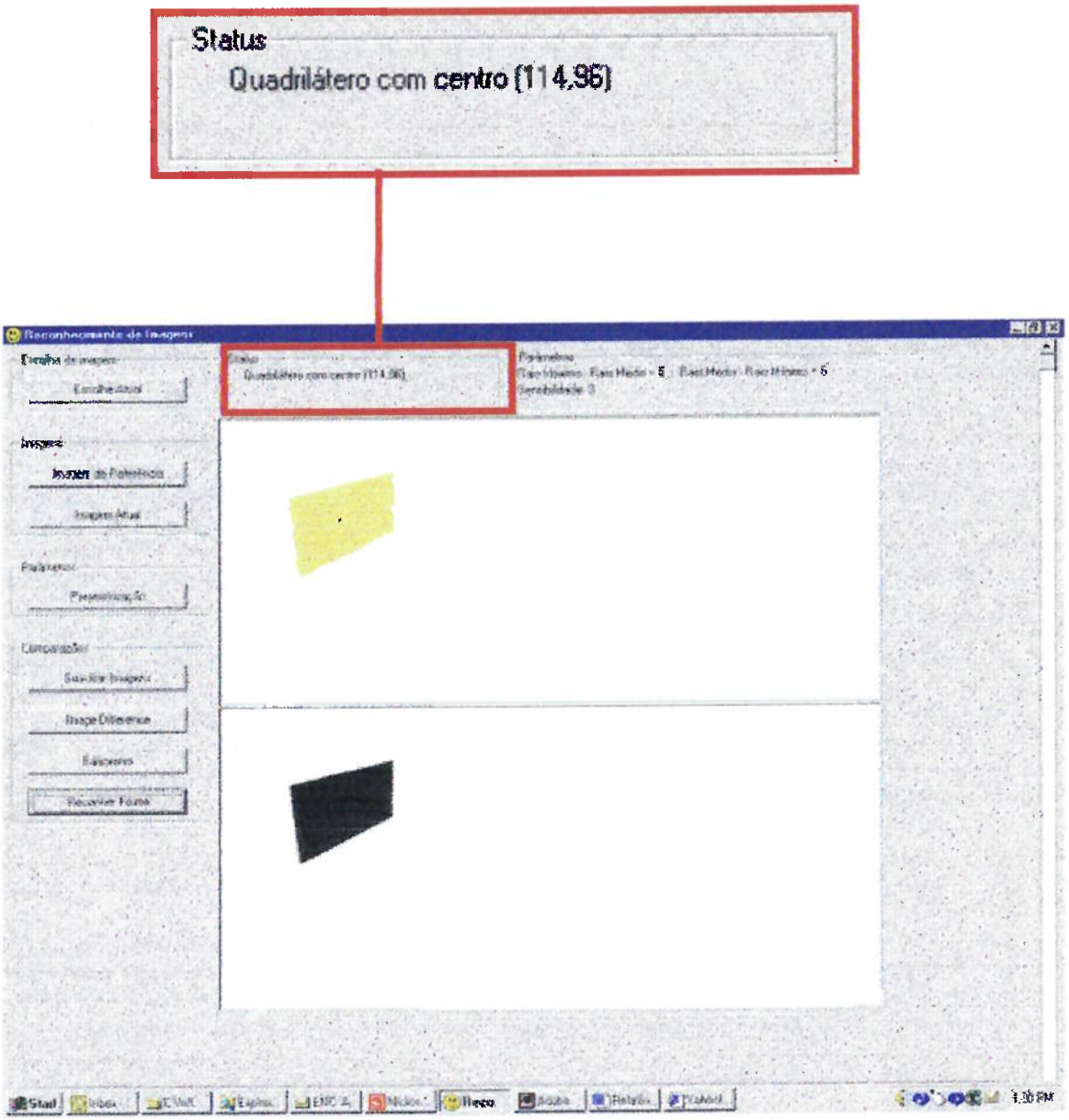
Outro Pentágono:



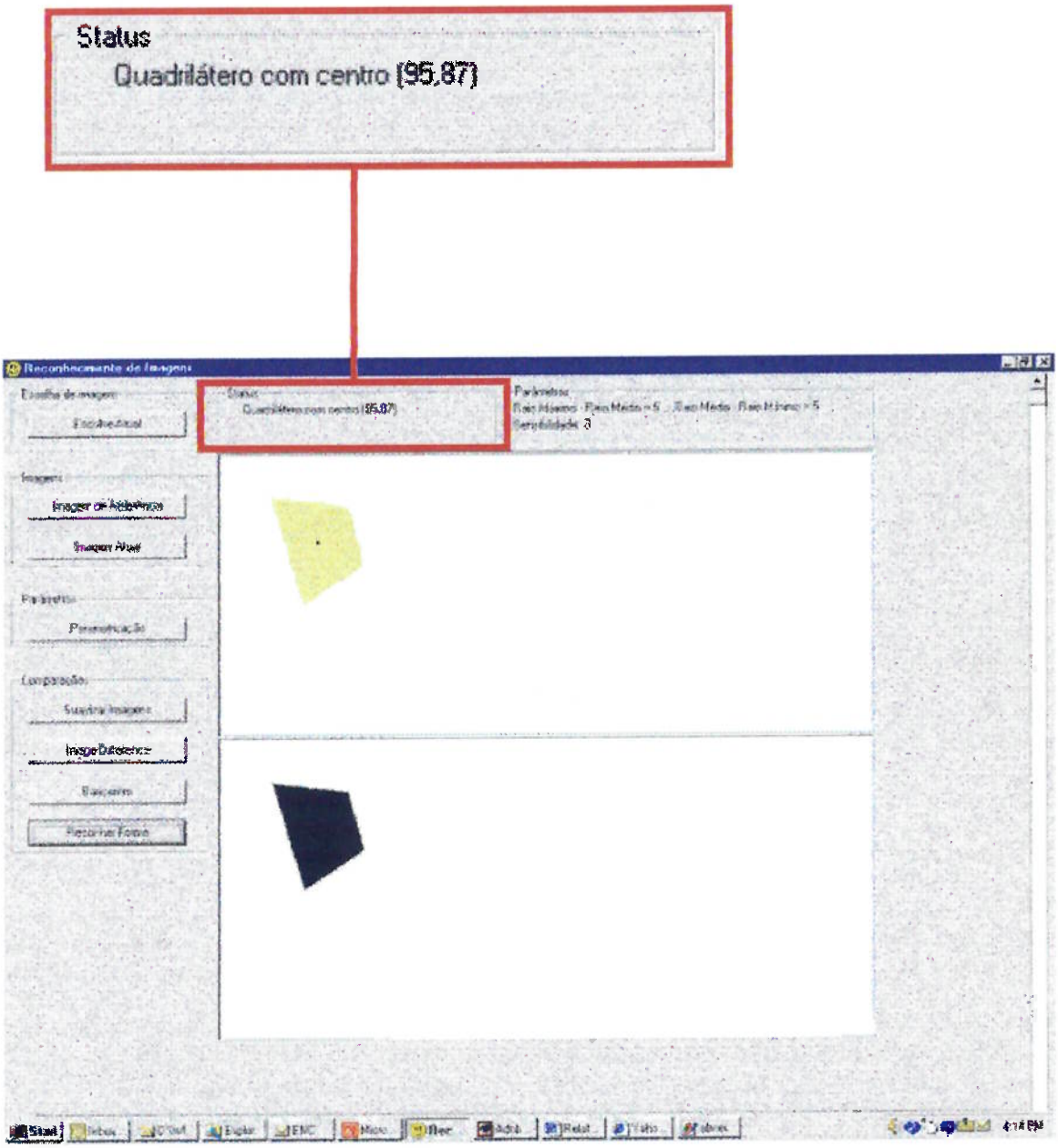
Um outro Quadrilátero:



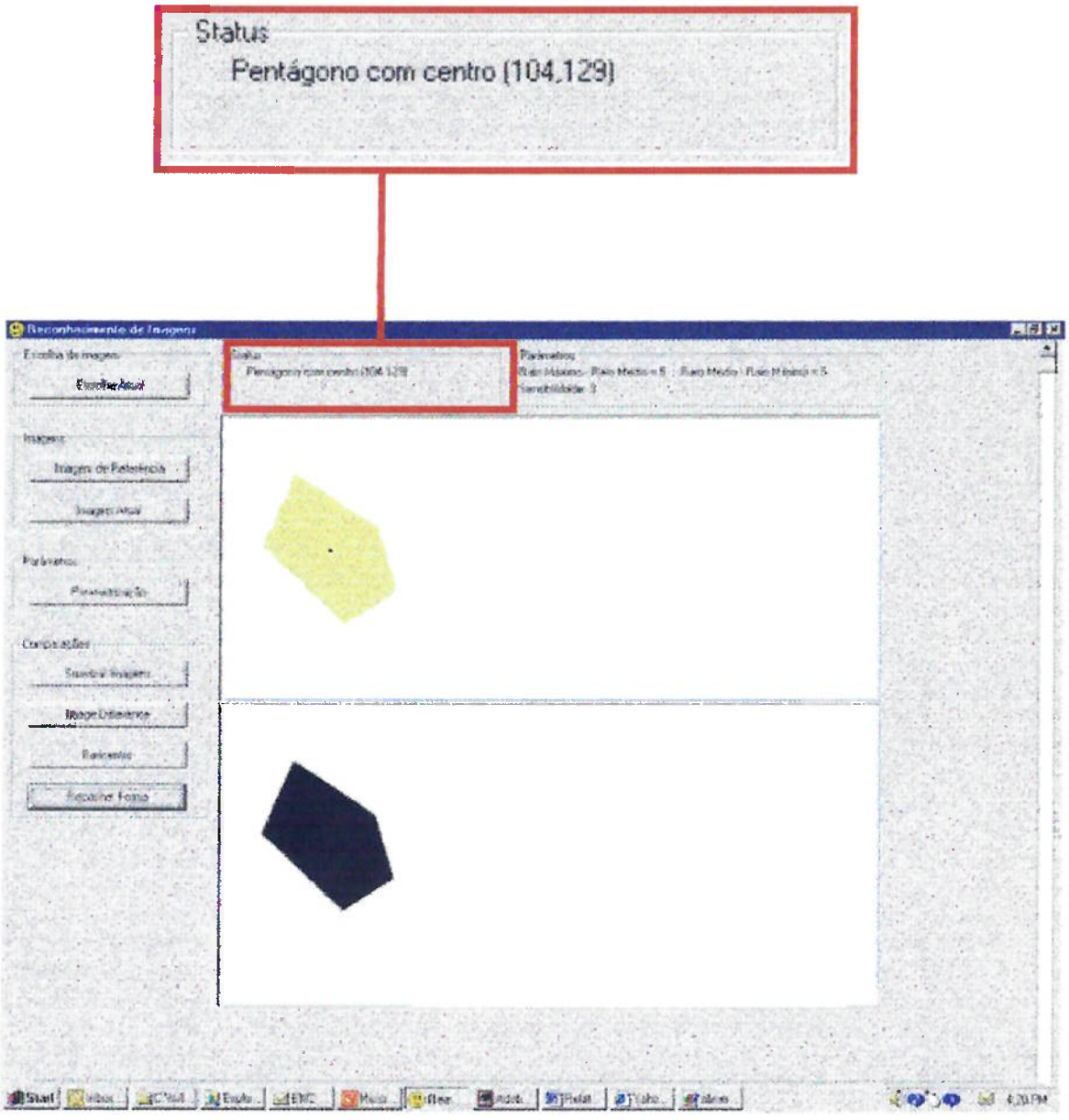
Outro Quadrilátero:



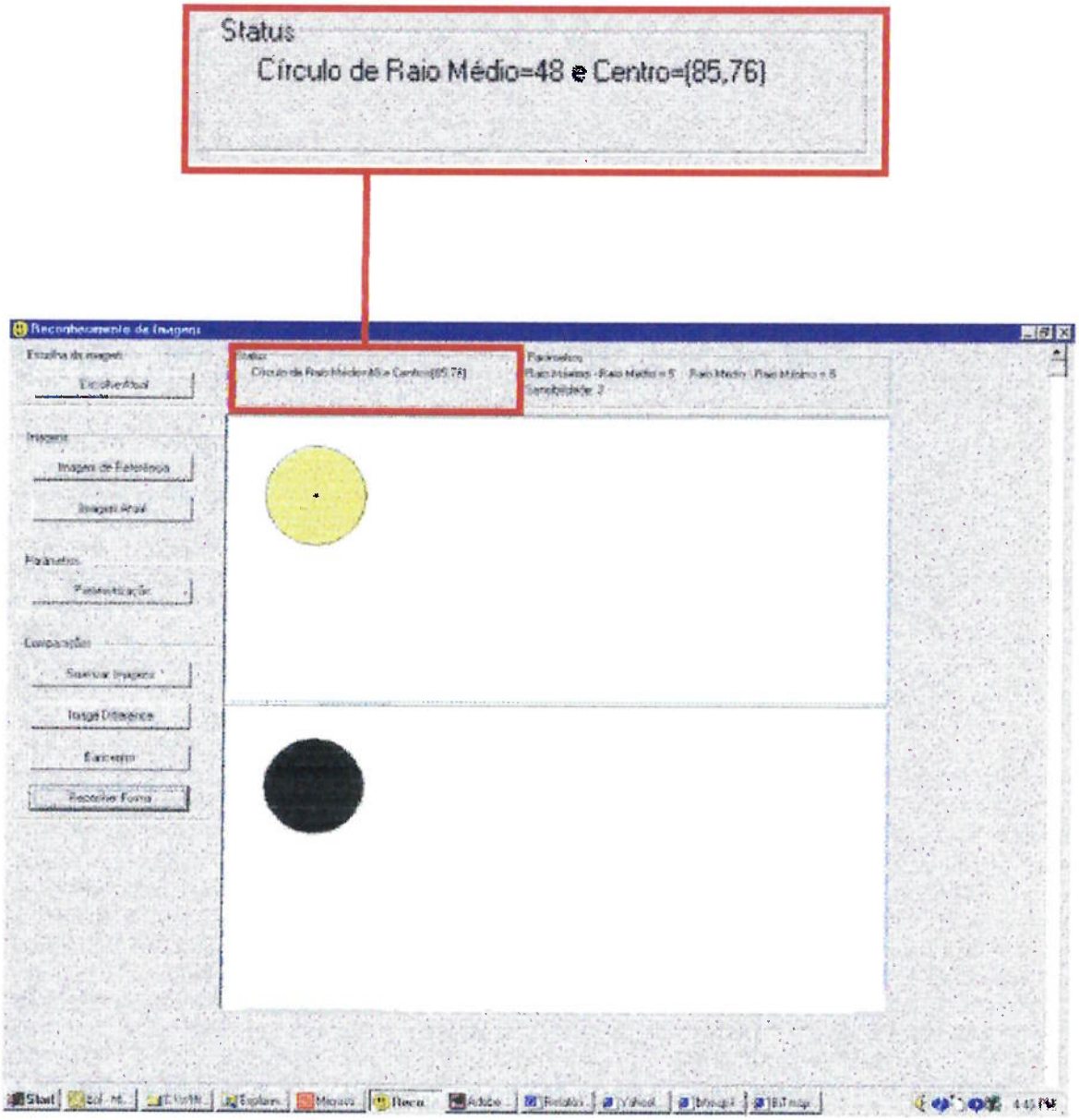
Mais um:



Outro pentágono:

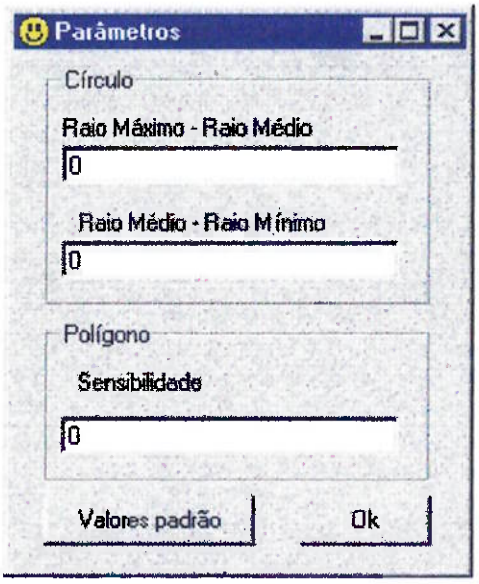


Nas próximas figuras serão mostrados exemplos de reconhecimento de círculos variando os parâmetros de comparação:

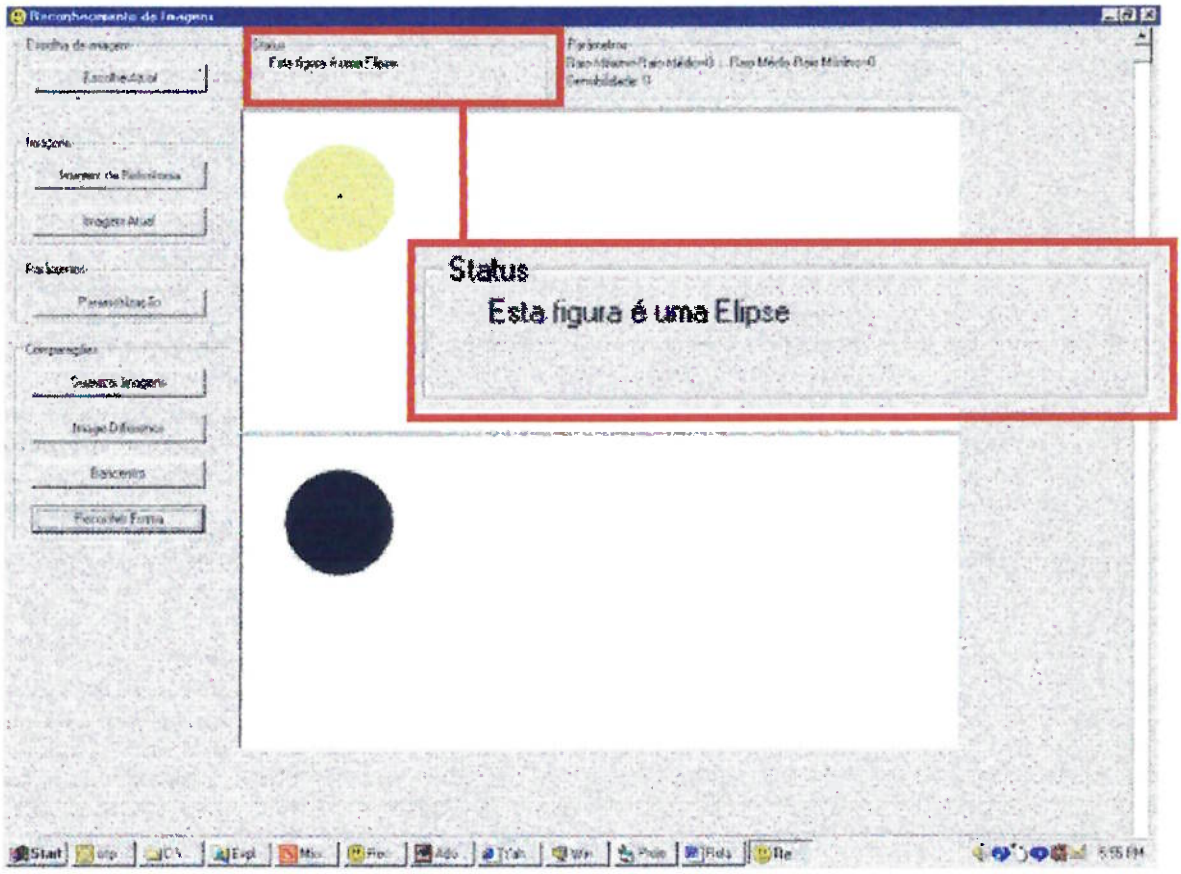


Esse círculo foi reconhecido utilizando valores padrão para comparação. Para exemplificar o funcionamento deste parâmetros será realizado o reconhecimento *setando* os valores para zero. Esta figura é uma elipse com excentricidade muito pequena e não será reconhecido como círculo, após será realizado outro processo de reconhecimento com uma elipse de excentricidade maior para verificar-se o influência dos parâmetros.

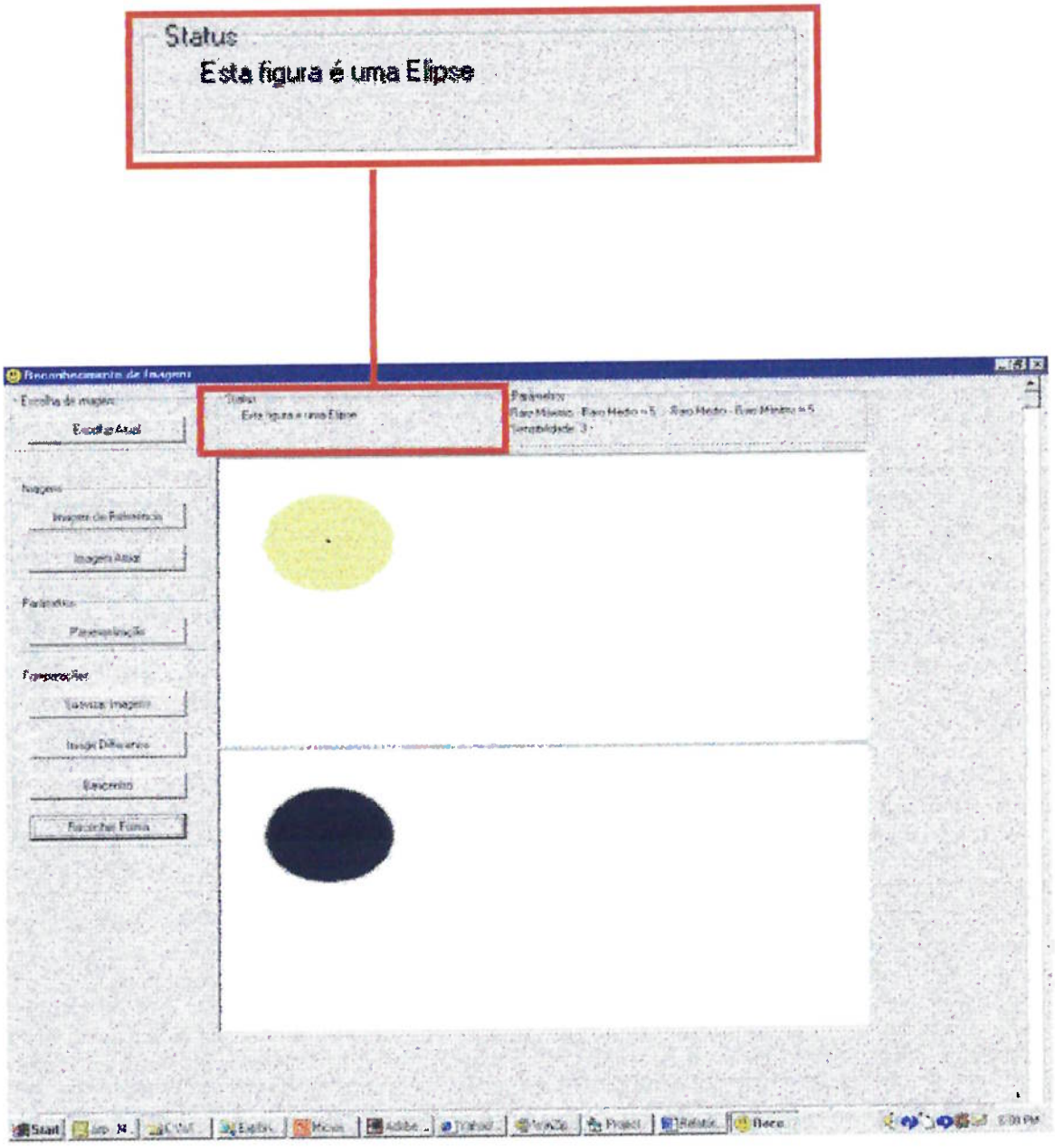
Parâmetros:



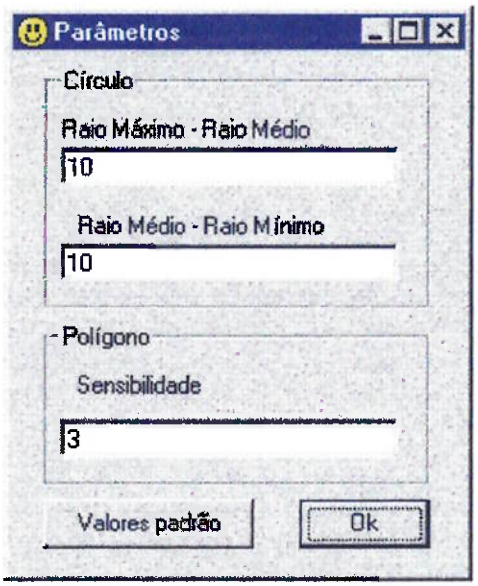
O reconhecimento:

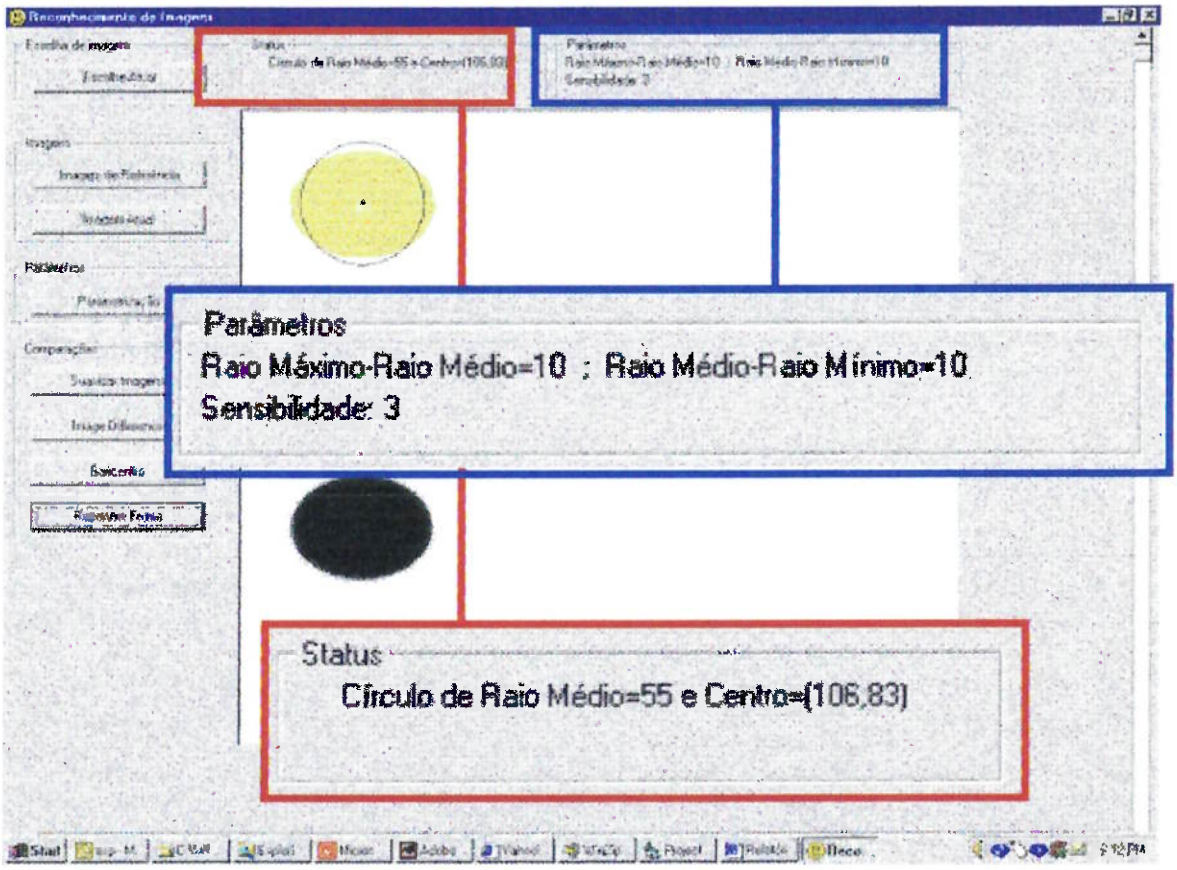


Como exemplo agora será usado uma elipse com excentricidade maior, com valores padrão para os parâmetros o software não reconhece o figura como círculo.



Alterando os parâmetros de maneira que a elipse seja reconhecida como um círculo:





Com os exemplos acima determina-se exatamente o objetivo da implementação destes algoritmos, ou seja, reconhecer formas de maneira que possam ser aplicadas na indústria de uma maneira geral, seja em funções de segurança ou em processos de automação industrial.

CONCLUSÃO E COMENTÁRIOS

Para o desenvolvimento deste software e dos respectivos algoritmos foram utilizadas metodologias ensinadas durante o curso Metodologia de Projeto o que facilitou o andamento do trabalho, metodologias de desenvolvimento de algoritmos vistos durante os cursos e laboratórios de computação também foram aplicados até o término do projeto.

O sistema apresentado é capaz de identificar uma alteração no ambiente que está monitorando, bem como procurar identificar uma forma básica nessa alteração. A aproximação do objeto por uma forma básica pode ser controlado pelo usuário através da definição dos parâmetros de comparação (por exemplo, uma elipse de baixa excentricidade pode ou não ser considerada um círculo, de acordo com especificações do usuário).

A utilização de algoritmos que se utilizam de propriedades geométricas das figuras simples é de grande vantagem sobre métodos que se utilizariam de banco de dados dessas imagens, evitando-se os problemas de escala e orientação no espaço da figura, assim o reconhecimento torna-se mais rápido e eficaz porque não é necessário tantos tratamentos para descobrir a relação de escala e mudança na orientação para alinhar as figuras.

Para o desenvolvimento dos algoritmos foram definidas algumas premissas básicas e que devem ser respeitadas na escolha das figuras a serem

reconhecidas, caso essas premissas não sejam respeitadas não há garantia de que o software irá funcionar adequadamente ou simplesmente não funciona.

As premissas são:

- Figuras simples;
- Figuras não concavidades, ou seja, sem reentrâncias;
- Figuras não furadas;
- Na área de trabalho não podem haver mais que uma figura;

As cores da figura não influenciam no reconhecimento porém a sensibilidade definida pelo usuário deve prever a alteração de cor no quadro sem que isso prejudique o reconhecimento, os valores padrão pré definidos prevêem este fato.

Uma vez que os algoritmos foram desenvolvidos de maneira inovadora, os valores padrões utilizados e também a questão da definição do número de vizinhos para determinar se aquele pixel é um vértice, um possível vértice ou não é um vértice foram definidos empiricamente o que nos expôs a uma situação de pesquisa e validação de dados, o que foi muito interessante para nós.

O sistema é de fácil implementação, utilizando uma câmera, uma placa digitalizadora e um microcomputador, além do software desenvolvido para essa aplicação. O fato de ser gerenciado por um microcomputador permite a integração desse sistema com demais sistemas ou outros sistemas de um edifício inteligente.

BIBLIOGRAFIA

[1] MASSARANI, Marcelo e outros . **PMC 475 Metodologia de Projeto** - Apostila do curso.

[2] MASCARENHAS, N.; VELASCO F. **Processamento de Imagens IV** - Escola de Computação, IMEUSP São Paulo, 1984.

[3] MOURA, L.; MARTINS, F. **Edge Detection through Cooperation and Competion** - Universidade de Londres, Tese de Doutorado, Londres 1988.

[4] HILDRETH, E. **Edge Detection** Boston MA, Massachussetts Institute of tecnology, September 1985. (A.I. Memo Number 858).

Sites:

- Adobe Solutions Network
- Bitmap Format (_bmp)
- bitmap from FOLDOC
- Bitmap Functions
- BMP- Windows Bitmap Format
- The BitMap Format

Anexo A

Código

- *Janela Principal:*

```
Dim ImagePixelsReferencia(2, 650, 500) As Integer
Dim ImagePixelsDiferentes(650, 500) As Integer
Dim vetor_de_vértices(650, 500) As Integer
Dim Ig, Jg, RM As Integer
Dim ImagePixelsAtual(2, 650, 500) As Integer
Public comp1, comp2, comp3 As Integer
Dim npossivelvertices, nvertices As Integer
Dim possivelvertice_i(500), possivelvertice_j(500) As Integer
Dim vertice_i(20), vertice_j(20) As Integer
Dim posicao As Integer
Dim primo As Boolean
Dim primoi, primoj, ultimoi, ultimoj As Integer
Dim i, j, R1, R2, R3, R4, R5, R6, R7, R8 As Integer

Private Sub botao_atual_Click()
Dim i, j
Dim red As Integer, green As Integer, blue As Integer
Dim pixel&

Picture2.Picture = LoadPicture(Trim(Form2.Text1.Text) & ".bmp")
For i = 1 To 280
  For j = 1 To 639
   pixel& = Form1.Picture2.Point(j, i)
```

```
    red = pixel& Mod 256
    green = ((pixel& And &HFF00) / 256&) Mod 256&
    blue = (pixel& And &HFF0000) / 65536

    ImagePixelsAtual(0, j, i) = red
    ImagePixelsAtual(1, j, i) = green
    ImagePixelsAtual(2, j, i) = blue
     Next
Next
Label1.Caption = "Matriz da Imagem Atual Montada"
End Sub

Private Sub botao_baricentro_Click()
Dim i, j
Dim a, di, dj

iA = 0
jA = 0
di = 0
dj = 0
For i = 10 To 250
  For j = 10 To 600
      If (ImagePixelsDiferentes(j, i) = 1) Then
          di = di + 1 * i
          dj = dj + 1 * j
          a = a + 1

    End If
```

```
    Next
Next
 Ig = Int(di / a)
 Jg = Int(dj / a)
 For i = Ig - 1 To Ig + 1
 For j = Jg - 1 To Jg + 1
 Picture1.PSet (j, i), RGB(0, 0, 0)
Next
Next
Label1.Caption = "Baricentro = " & "(" & Jg & "," & Ig & ")"

End Sub

Function distancia(xa, ya, xb, yb) As Integer

 distancia = Sqr((xa - xb) * (xa - xb) + (ya - yb) * (ya - yb))
 MsgBox "distancia=" & distancia

End Function

Function Existepossivelvertice(x, y) As Integer

Dim k, posi As Integer
Dim achou As Boolean

For k = 1 To npossivelvertices
  If (x = possivelvertice_i(k) And y = possivelvertice_j(k)) Then
    achou = True
```

```
    posi = k
  End If
Next

  If (achou = True) Then
      Existepossivelvertice = posi
  Else

      Existepossivelvertice = 0
        End If

End Function

Private Sub botao_diferenca_Click()
Dim i, j
For i = 1 To 280
  For j = 1 To 639
      ImagePixelsDiferentes(j, i) = 0
       If  (Abs(ImagePixelsAtual(0,  j,  i)  -  ImagePixelsReferencia(0,  j,
i)) > 40) Then
      ImagePixelsDiferentes(j, i) = 1
    End If
    If (Abs(ImagePixelsAtual(1, j, i) - ImagePixelsReferencia(1, j, i)) >
40) Then
      ImagePixelsDiferentes(j, i) = 1
    End If
        If  (Abs(ImagePixelsAtual(2,  j,  i)  -  ImagePixelsReferencia(2,  j,
i)) > 60) Then
```

```
      ImagePixelsDiferentes(j, i) = 1
    End If
  Next
Next
For i = 10 To 250
  For j = 10 To 600
     If ImagePixelsDiferentes(j, i) = 0 Then
      Picture1.PSet (j, i), RGB(255, 255, 255)
    End If

    If ImagePixelsDiferentes(j, i) = 1 Then
      Picture1.PSet (j, i), RGB(240, 250, 150)
    End If

    If ImagePixelsDiferentes(j, i) = 2 Then
      Picture1.PSet (j, i), RGB(100, 50, 190)
    End If

    If ImagePixelsDiferentes(j, i) = 3 Then
      Picture1.PSet (j, i), RGB(0, 0, 0)
    End If
   Next
Next
Label1.Caption = "Diferenças entre as imagens montada"
End Sub

Private Sub botao_escolha_Click()
Load Form2
```

```
Form2.Show
End Sub

Function Achavertice(xa, ya) As Boolean
Dim aux, var, xi, yi, dist, a, b, k As Integer
Dim achou As Boolean

aux = 0
achou = False

MsgBox "entrou no achavertice"
For var = 1 To nvertices
   xi = vertice_i(var)
   yi = vertice_j(var)
   dist = distancia(xa, ya, xi, yi)
   If (dist > aux) Then
      aux = dist
   End If
Next

For var = 1 To npossivelvertices
   xi = possivelvertice_i(var)
   yi = possivelvertice_j(var)
   dist = distancia(xa, ya, xi, yi)
   If (dist > aux) Then
    aux = dist
    achou = True
    posicao = var
```

```
    a = xi
    b = yi
   End If
 Next

If (achou = True) Then
  nvertices = nvertices + 1
  vertice_i(nvertices) = a
  vertice_j(nvertices) = b
    For k = posicao To npossivelvertices - 1
    possivelvertice_i(k) = possivelvertice_i(k + 1)
    possivelvertice_j(k) = possivelvertice_j(k + 1)
  Next
  npossivelvertices = npossivelvertices - 1


End If
MsgBox "saindo do achavertice, achou =" & achou
Achavertice = achou

End Function

Function sinal(p)
If p < 0 Then
  sinal = -1
Else
  sinal = 1
  End If
```

```
End Function

Function Limpaentredoispontos() As Integer

Dim k, l, xa, xb, ya, yb, a, b, i, j, v, c, ka As Integer

For k = 1 To nvertices
 For l = 1 To nvertices

If (k <> l) Then
xa = vertice_i(k)
ya = vertice_j(k)
xb = vertice_i(l)
yb = vertice_j(l)
m = (yb - ya) / (xb - xa)
a = xa
b = ya
i = xb - xa
j = yb - ya

For v = 1 To Abs(i) - 1
  a = xa + v * sinal(i)
  b = ya + m * v
  For c = (b - 3) To (b + 3)
    pos = Existepossivelvertice(a, c)
    If (pos > 0) Then
    For ka = pos To npossivelvertices - 1
```

```
    possivelvertice_i(ka) = possivelvertice_i(ka + 1)
    possivelvertice_j(ka) = possivelvertice_j(ka + 1)
    Next
  npossivelvertices = npossivelvertices - 1

    End If
  Next
Next

For v = 1 To Abs(j) - 1
  b = yb + v * sinal(j)
  a = xb + 1 / m * v
  For c = (a - 3) To (a + 3)
      pos = Existepossivelvertice(c, b)
      If (pos > 0) Then
      For ka = pos To npossivelvertices - 1
     possivelvertice_i(ka) = possivelvertice_i(ka + 1)
     possivelvertice_j(ka) = possivelvertice_j(ka + 1)
     Next
     npossivelvertices = npossivelvertices - 1
    End If
  Next
Next
End If
Next
Next
 Limpaentredoispontos = 0
```

```
End Function

Private Sub botao_forma_Click()

Dim Fim, achou As Boolean
Dim Raiomax, Raiomin, nvizi, k, esquece As Integer

achou = False
nvertices = 0
npossivelvertices = 0

R1 = 0
R2 = 0
R3 = 0
R4 = 0
R5 = 0
R6 = 0
R7 = 0
R8 = 0

Fim = False
i = Ig
j = Jg

Do
    j = j + 1
        If ImagePixelsDiferentes(j, i) = 1 Then
       R1 = R1 + 1
```

```
     Else
         Fim = True
    End If
Loop While Fim = False
Fim = False
i = Ig
j = Jg

Do
    i = i + 1
    j = j + 1
   If ImagePixelsDiferentes(j, i) = 1 Then
                R2 = R2 + 1
    Else
         Fim = True
    End If
Loop While Fim = False

Fim = False
i = Ig
j = Jg

Do
    i = i + 1
   If ImagePixelsDiferentes(j, i) = 1 Then
         R3 = R3 + 1
     Else
         Fim = True
```

```
    End If
Loop While Fim = False
Fim = False
i = Ig
j = Jg

Do
    i = i + 1
    j = j - 1
   If ImagePixelsDiferentes(j, i) = 1 Then
       R4 = R4 + 1
     Else
        Fim = True
    End If
Loop While Fim = False

Fim = False
i = Ig
j = Jg

Do
    j = j - 1
    If ImagePixelsDiferentes(j, i) = 1 Then
       R5 = R5 + 1
     Else
        Fim = True
    End If
Loop While Fim = False
```

```
Fim = False
i = Ig
j = Jg

Do
    i = i - 1
    j = j - 1
    If ImagePixelsDiferentes(j, i) = 1 Then
          R6 = R6 + 1
     Else
        Fim = True
    End If

Loop While Fim = False

Fim = False
i = Ig
j = Jg

Do
    i = i - 1
    If ImagePixelsDiferentes(j, i) = 1 Then
       R7 = R7 + 1
     Else
        Fim = True
    End If
  Loop While Fim = False
```

```
Fim = False
i = Ig
j = Jg

Do
    i = i - 1
    j = j + 1
   If ImagePixelsDiferentes(j, i) = 1 Then
          R8 = R8 + 1
     Else
        Fim = True
    End If
Loop While Fim = False

R2 = Sqr(2) * R2
R4 = Sqr(2) * R4
R6 = Sqr(2) * R6
R8 = Sqr(2) * R8

RM = (R1 + R2 + R3 + R4 + R5 + R6 + R7 + R8) / 8

Raiomax = Maximo(R1, R2, R3, R4, R5, R6, R7, R8)
Raiomin = Minimo(R1, R2, R3, R4, R5, R6, R7, R8)
```

```
If Form3.Text4.Text = 0 Then
  comp1 = Form3.Text1.Text
  comp2 = Form3.Text2.Text
  comp3 = Form3.Text3.Text
Else
  comp1 = 5
  comp2 = 5
  comp3 = 3
End If

If (Raiomax - RM) <= comp1 And (RM - Raiomin) <= comp2 Then
    Picture1.Circle (Jg, Ig), RM, RGB(0, 0, 0)
    Label1.Caption = "Círculo de Raio Médio=" & RM & " e Centro=(" & Jg &
"," & Ig & ")"
Else

  nvertices = 0
  npossivelvertices = 0
  primo = False

  For j = 2 To 300
   For i = 2 To 260

     If (ImagePixelsDiferentes(j, i) = 1) Then
     If (primo = False) Then
        primoj = j
        primoi = i
        primo = True
```

```
      End If

      ultimoi = i
      ultimoj = j

      nvizi = verifica_vizinhos(j, i)

      If (nvizi <= 3) Then
         nvertices = nvertices + 1
         vertice_i(nvertices) = i
         vertice_j(nvertices) = j
      Else
           If (nvizi = 4) Then
                npossivelvertices = npossivelvertices + 1
                possivelvertice_i(npossivelvertices) = i
               possivelvertice_j(npossivelvertices) = j
           End If
      End If
   End If
   Next
Next

MsgBox "Número de Vértices = " & nvertices

If (nvertices = 0) Then
   nvertices = 1
   vertice_i(nvertices) = primoi
   vertice_j(nvertices) = primoj
```

```
End If
If (nvertices = 1) Then
   nvertices = 1
   vertice_i(nvertices) = ultimoi
   vertice_j(nvertices) = ultimoj
End If

Do
  For k = 1 To nvertices

  achou = Achavertice(vertice_i(k), vertice_j(k))
  esquece = Limpaentredoispontos()

Next
Loop While npossivelvertices > 0

MsgBox " Possíveis vértices = " & npossiveisvertices
End If

Label1.Caption = "Quantidade de Vértices: " & nvertices
Select Case nvertices
      Case 2
      Label1.Caption = "Dois" & "(" & Jg & "," & Ig & ")"
      Case 3
      Label1.Caption = "Triângulo com centro " & "(" & Jg & "," & Ig &
")"
      Case 4
```

```
        Label1.Caption = "Quadrilátero com centro " & "(" & Jg & "," & Ig &
")"
        Case 5
        Label1.Caption = "Pentágono com centro " & "(" & Jg & "," & Ig &
")"
        Case 6
        Label1.Caption = "Hexágono com centro " & "(" & Jg & "," & Ig & ")"
        Case 7
        Label1.Caption = "Heptágono com centro " & "(" & Jg & "," & Ig &
")"
        Case 8
        Label1.Caption = "Octógono com centro " & "(" & Jg & "," & Ig & ")"
        Case 9
        Label1.Caption = "Eneágono com centro " & "(" & Jg & "," & Ig & ")"
        Case 10
        Label1.Caption = "Decágono com centro " & "(" & Jg & "," & Ig & ")"
        Case Else
        Label1.Caption = "Esta figura é uma Elipse"
End Select

End Sub

Private Sub botao_parametros_Click()
Load Form3
Form3.Show
End Sub

Private Sub botao_referencia_Click()
```

```
Dim i, j
Dim red As Integer, green As Integer, blue As Integer
Dim pixel&

Picture1.Picture = LoadPicture("abner01.bmp")

For i = 1 To 280
  For j = 1 To 639

    pixel& = Form1.Picture1.Point(j, i)

    red = pixel& Mod 256
    green = ((pixel& And &HFF00) / 256&) Mod 256&
    blue = (pixel& And &HFF0000) / 65536

    ImagePixelsReferencia(0, j, i) = red
    ImagePixelsReferencia(1, j, i) = green
    ImagePixelsReferencia(2, j, i) = blue

  Next
Next

Label1.Caption = "Matriz da Imagem Referência Montada"

End Sub

Private Sub botao_suavizar_Click()
```

```
Dim i, j
Dim verde, vermelho, azul As Long

verde = 0
vermelho = 0
azul = 0

 Load Form4
 Form4.Show
 Form4.ProgressBar1.Min = 3
 Form4.ProgressBar1.Max = 276

For i = 3 To 276
 For j = 3 To 636

   vermelho = (ImagePixelsReferencia(0, j - 1, i - 1) +
ImagePixelsReferencia(0, j, i - 1) + ImagePixelsReferencia(0, j + 1, i -
1) + ImagePixelsReferencia(0, j - 1, i) + ImagePixelsReferencia(0, j, i)
+ ImagePixelsReferencia(0, j + 1, i) + ImagePixelsReferencia(0, j - 1, i
+ 1) + ImagePixelsReferencia(0, j, i + 1) + ImagePixelsReferencia(0, j +
1, i + 1)) / 9
   verde = (ImagePixelsReferencia(1, j - 1, i - 1) +
ImagePixelsReferencia(1, j, i - 1) + ImagePixelsReferencia(1, j + 1, i -
1) + ImagePixelsReferencia(1, j - 1, i) + ImagePixelsReferencia(1, j, i)
+ ImagePixelsReferencia(1, j + 1, i) + ImagePixelsReferencia(1, j - 1, i
+ 1) + ImagePixelsReferencia(1, j, i + 1) + ImagePixelsReferencia(1, j +
1, i + 1)) / 9
```

```
   azul = (ImagePixelsReferencia(2, j - 1, i - 1) +
ImagePixelsReferencia(2, j, i - 1) + ImagePixelsReferencia(2, j + 1, i -
1) + ImagePixelsReferencia(2, j - 1, i) + ImagePixelsReferencia(2, j, i)
+ ImagePixelsReferencia(2, j + 1, i) + ImagePixelsReferencia(2, j - 1, i
+ 1) + ImagePixelsReferencia(2, j, i + 1) + ImagePixelsReferencia(2, j +
1, i + 1)) / 9

   Picture1.PSet (j, i), RGB(vermelho, verde, azul)

   pixel& = Form1.Picture1.Point(j, i)

   red = pixel& Mod 256
   green = ((pixel& And &HFF00) / 256&) Mod 256&
   blue = (pixel& And &HFF0000) / 65536

   ImagePixelsReferencia(0, j, i) = red
   ImagePixelsReferencia(1, j, i) = green
   ImagePixelsReferencia(2, j, i) = blue

   vermelho = (ImagePixelsAtual(0, j - 1, i - 1) + ImagePixelsAtual(0, j,
i - 1) + ImagePixelsAtual(0, j + 1, i - 1) + ImagePixelsAtual(0, j - 1,
i) + ImagePixelsAtual(0, j, i) + ImagePixelsAtual(0, j + 1, i) +
ImagePixelsAtual(0, j - 1, i + 1) + ImagePixelsAtual(0, j, i + 1) +
ImagePixelsAtual(0, j + 1, i + 1)) / 9
```

```
   verde = (ImagePixelsAtual(1, j - 1, i - 1) + ImagePixelsAtual(1, j, i
- 1) + ImagePixelsAtual(1, j + 1, i - 1) + ImagePixelsAtual(1, j - 1, i)
+ ImagePixelsAtual(1, j, i) + ImagePixelsAtual(1, j + 1, i) +
ImagePixelsAtual(1, j - 1, i + 1) + ImagePixelsAtual(1, j, i + 1) +
ImagePixelsAtual(1, j + 1, i + 1)) / 9
   azul = (ImagePixelsAtual(2, j - 1, i - 1) + ImagePixelsAtual(2, j, i -
1) + ImagePixelsAtual(2, j + 1, i - 1) + ImagePixelsAtual(2, j - 1, i) +
ImagePixelsAtual(2, j, i) + ImagePixelsAtual(2, j + 1, i) +
ImagePixelsAtual(2, j - 1, i + 1) + ImagePixelsAtual(2, j, i + 1) +
ImagePixelsAtual(2, j + 1, i + 1)) / 9

   Picture2.PSet (j, i), RGB(vermelho, verde, azul)

   pixel& = Form1.Picture2.Point(j, i)

   red = pixel& Mod 256
   green = ((pixel& And &HFF00) / 256&) Mod 256&
   blue = (pixel& And &HFF0000) / 65536

   ImagePixelsAtual(0, j, i) = red
   ImagePixelsAtual(1, j, i) = green
   ImagePixelsAtual(2, j, i) = blue

   Form4.ProgressBar1.Value = i
 Next
Next
Form4.Hide
Unload Form4
```

```
Label1.Caption = "Imagens Suavizadas"

End Sub

Function Maximo(n1, n2, n3, n4, n5, n6, n7, n8 As Integer) As Integer

If (n1 > n2) Then
rmax = n1
Else
rmax = n2
End If

If (n3 > rmax) Then
rmax = n3
End If

If (n4 > rmax) Then
rmax = n4
End If

If (n5 > rmax) Then
rmax = n5
End If

If (n6 > rmax) Then
rmax = n6
End If
```

```
If (n7 > rmax) Then
rmax = n7
End If


If (n8 > rmax) Then
rmax = n8
End If

Maximo = rmax
End Function

Function Minimo(n1, n2, n3, n4, n5, n6, n7, n8 As Integer) As Integer

If (n1 < n2) Then
rmin = n1
Else: rmin = n2
End If

If (n3 < rmin) Then
rmin = n3
End If

If (n4 < rmin) Then
rmin = n4
End If
```

```
If (n5 < rmin) Then
rmin = n5
End If

If (n6 < rmin) Then
rmin = n6
End If

If (n7 < rmin) Then
rmin = n7
End If

If (n8 < rmin) Then
rmin = n8
End If

Minimo = rmin

End Function

Function verifica_vizinhos(y, x) As Integer
Dim aux

    aux = 0

   If ImagePixelsDiferentes(j - 1, i - 1) = 1 Then
        aux = aux + 1
    End If
```

```
If ImagePixelsDiferentes(j, i - 1) = 1 Then
     aux = aux + 1
 End If

If ImagePixelsDiferentes(j + 1, i - 1) = 1 Then
     aux = aux + 1
 End If
If ImagePixelsDiferentes(j + 1, i) = 1 Then
     aux = aux + 1
 End If
If ImagePixelsDiferentes(j + 1, i + 1) = 1 Then
     aux = aux + 1
 End If

If ImagePixelsDiferentes(j, i + 1) = 1 Then
     aux = aux + 1
 End If

If ImagePixelsDiferentes(j - 1, i + 1) = 1 Then
     aux = aux + 1
 End If

If ImagePixelsDiferentes(j - 1, i) = 1 Then
     aux = aux + 1
 End If

 verifica_vizinhos = aux
```

```
End Function
```

Janela para Escolha da Imagem Atual:

```
Private Sub botao_ok_Click()

If Trim(Text1.Text) = "" Then
    MsgBox "Digite o nome da imagem", vbExclamation, "Atenção"
End If

If Trim(Text1.Text) <> "" Then
 Form2.Hide
End If

End Sub
```

Janela para Parametrização:

```
Private Sub botao_ok2_Click()

Text4.Text = 0
If Form3.Text1.Text = "" Or Form3.Text2.Text = "" Or Form3.Text3.Text =
"" Then
 MsgBox "Entre com todos os parâmetros", vbExclamation, "Atenção"
Else

Form1.Label2.Caption = "Raio Máximo-Raio Médio=" & Form3.Text1.Text & "
;  Raio Médio-Raio Mínimo=" & Form3.Text2.Text
Form1.Label3.Caption = "Sensibilidade: " & Form3.Text3.Text

Form3.Hide
End If

End Sub

Private Sub botao_padrao_Click()

Text4.Text = 1

Form1.Label2.Caption = "Raio Máximo - Raio Médio = " & 5 & "   ;   Raio
Médio - Raio Mínimo = " & 5
Form1.Label3.Caption = "Sensibilidade: " & 3
```

```
Form3.Hide

End Sub
```

Anexo B

Data Sheet da Placa de Aquisição

Member of MACH Series™

DT3153

Composite Color PCI Frame Grabber

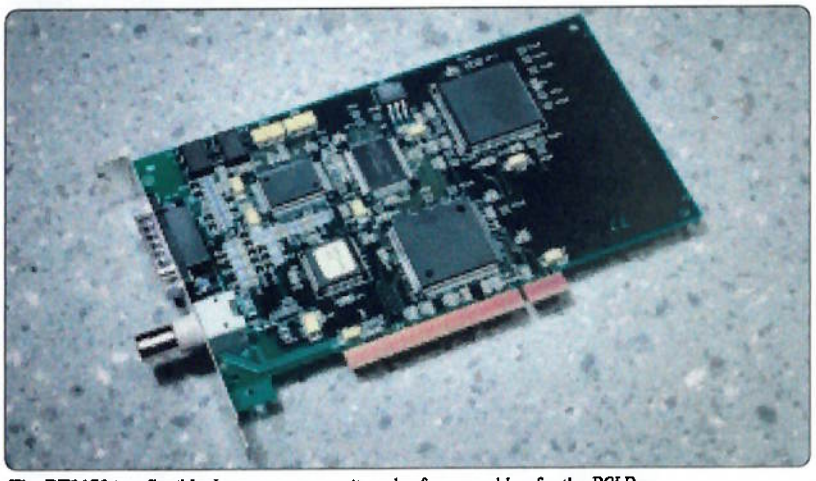
Key Features

- *Input flexibility supports up to three NTSC/PAL video sources or one S-video (Y/C) and two NTSC/PAL video sources.*
- *Sync Master mode supports multiple camera configurations.*
- *High-performance scaler enhances detail in small images*
- *General-purpose digital inputs/outputs for interfacing to peripheral devices*
- *MACH Series™ PCI bus-mastering architecture enables acquisition and transfer to memory at 30 fps (NTSC).*
- *Free DT-Acquire™ software enables you to capture, display, and save image data*

Compatible Windows Software

DT-Open Layers™

DT-Active Open Layers	32-Bit Frame Grabber SDK for Windows 95/98/NT	DT-Acquire
✔	✔	✔



The DT3153 is a flexible, low-cost, composite color frame grabber for the PCI Bus.

M-5853

Overview

The DT3153 is a composite color frame grabber for the PCI bus. The board can handle both composite and S-video signals, and can act as a sync master with compatible cameras. As a member of the MACH Series, it is a PCI Bus Master, capable of transferring images in real time to system memory.

32-bit True Color Format—All on the PCI Bus

The DT3153 captures composite or S-video images in NTSC and PAL formats, and passes the data directly onto the PCI bus. The image data can be output in 32-bit RGB, 15-bit RGB, or 16-bit YUV formats.

Added Camera Flexibility

Unlike other low-cost color frame grabbers, the DT3153 supports sync-master mode camera configurations. The DT3153 generates horizontal and vertical drive, as well as composite sync signals, to allow multiple cameras to be genlocked.

On-board Scaler

The DT3153 features a high-performance on-board scaler which can scale images from 100% down to 1% using linear phase interpolation, allowing small images to be viewed in great detail.

Ideal Applications

Visual Inspection
Color Machine Vision
Medical Imaging/Diagnostics
Surveillance/Security
Scientific Image Analysis

High Performance Data Transfer and Display

The DT3153 employs the industry-leading MACH Series architecture for real-time image display. Taking advantage of the PCI bus' high speed, up to 132 MB/s, the DT3153 can transfer an unlimited number of frames in real time across the bus to host memory. Using the DirectDraw™ (DDI) standard built into Windows 95/98 and Windows NT 4.0, you can display live video with non-destructive overlays without having display hardware on the frame grabber. And by using a separate VGA card for display, you are free to choose the graphics card capabilities that satisfy your particular application needs.

DT3153 (MACH Series™)

BUS: PCI

Type: Composite Color

System CPU Free for Image Processing

Because system resources are not involved in transferring data with the DT3153 Bus Master design, your computer's CPU is free to perform high-speed image processing on the data you acquire. You can acquire a second image while using the host CPU to process the first.

Extensive Software Support Saves Programming Time, Protects Your Investment

Several software options are available to help you get your application up and running quickly and easily. DT-Active Open Layers™ is an ActiveX® control that enables you to use Data Translation's PCI frame grabbers with graphical programming environments such as Microsoft Visual Basic and Visual C++. The Frame Grabber SDK™ is a complete library of hardware-independent function calls that enables you to control the operations of Data Translation's PCI frame grabbers in Visual C and Visual C++.

Both packages adhere to Data Translation's DT-Open Layers® software architecture, which provides a common application programming interface (API) across all supported boards. This means that you can easily switch from one Data Translation frame grabber to another, or add more frame grabbers, with little or no reprogramming. Adding support for a new board is as easy as installing a new driver.

DT3153 Input Flexibility

The DT3153 is designed for ease-of-use and input flexibility. It can support up to three NTSC/PAL (composite) video sources, or one Y/C (S-video) input and two NTSC/PAL video sources. In addition, the DT3153 has a single-input BNC connector for use with a single composite source.

Technical Support

As you develop your application, technical support is available when you need it. Extensive information is available 24 hours a day on our web site at www.datatranslation.com, including drivers, example code, bug fixes, pinouts, a searchable KnowledgeBase, and much more.

Support is also available from your point of purchase. Telephone support is free for the first 90 days; you can also request complimentary support via e-mail or fax at any time. Additional support options are available; contact your Data Translation representative for details.

Real-Time Display, Non-Destructive Overlays

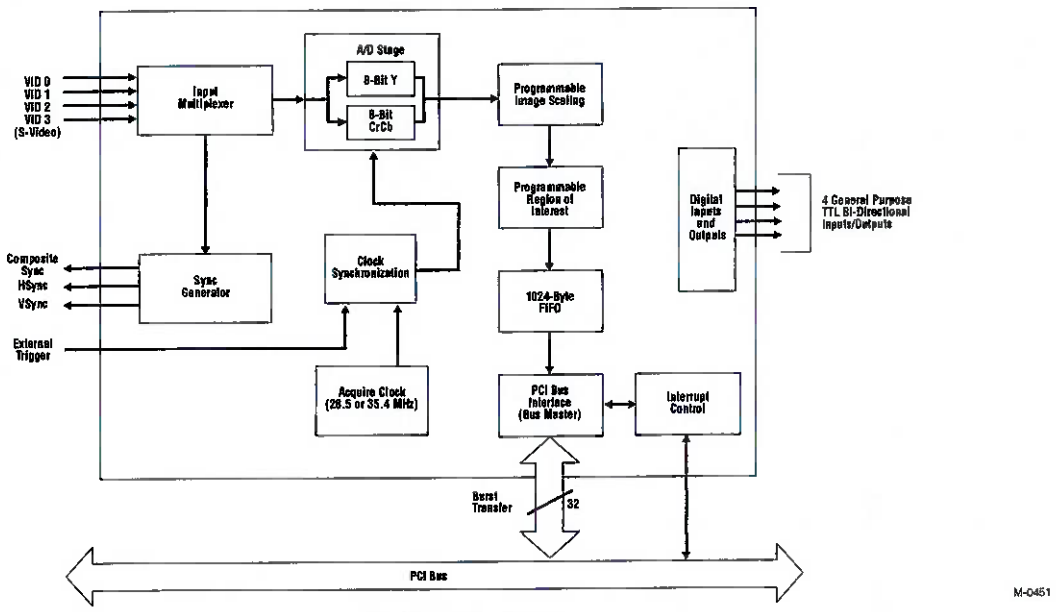
MACH Series frame grabbers employ Microsoft's DirectDraw (DDI) standard, allowing you to display real-time, live video with non-destructive overlays without adding costly display hardware (i.e. VGA circuitry) to the frame grabber. This approach offers many advantages over traditional frame grabber display and overlay methods, including:

Minimal CPU Bandwidth: The DirectDraw display technique requires minimal CPU bandwidth, leaving the CPU free to perform image processing or other tasks. Ideal for applications where display video and processing occur simultaneously, DDI allows for stagger-free images and smooth flowing, real-time video with overlays.

Upgradable Compatibility: With DDI, your MACH Series frame grabber will work with any DirectDraw-compatible graphics card. And since DirectDraw is enabled through the graphics card driver, you can upgrade an existing graphics card to DDI by simply loading a new driver.

Flexible Graphics Card Selection: Because the graphics card is not built onto the frame grabber, you are not "locked in" to the performance of the frame grabber's display circuitry. This allows you to choose the frame grabber that suits your needs and the graphics card that meets your performance requirements and budget.

Additional Features: Since DDI is the same overlay technique used by video game manufacturers, this capability gives you the ability to have non-destructive overlays of any size, shape, or color on top of live video. In addition, overlays can be translucent (semi clear), rotated, animated, or even placed over scaled images.

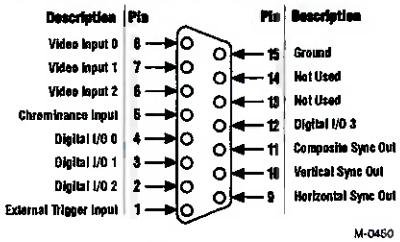


DT3153 Block Diagram

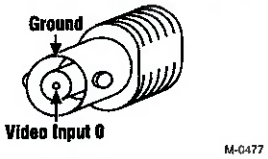
User Connections

Connecting S-Video Signals to the DT3153

To connect S-video input sources to the DT3153 frame grabber, you need to purchase an EP306 cable. Additional items that are needed include: one S-video (male) mating connector, and two solderable male BNC adapters. These can be purchased at your local electronics store.



Connector J1



Onboard BNC Connector for Single Video Input

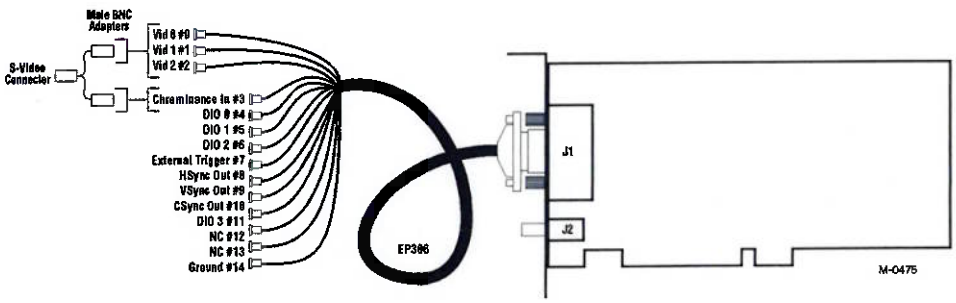


Figure 1. Connecting the EP306 Cable to Connector J1

DT3153 (MACH Series™)

BUS: PCI

Type: Composite Color

DT3153 Specifications

All specifications are typical at +25 °C and rated voltage unless otherwise specified.

Video Input

Video Format: Composite video and S-Video (Y/C) formats; NTSC (60 Hz), PAL (50 Hz); software selectable

Timing Format: Standard 60 Hz and 50 Hz format timing supported; software selectable

Inputs: 3 NTSC/PAL inputs or 1 S-Video (Y/C) and 2 NTSC/PAL inputs; ac coupled

Video Signal: 1 Volt peak-to-peak, 75 ohms

Spatial Resolution: 640 x 480 (60 Hz), 768 x 576 (50 Hz)

Acquisition

Digitization: Twin 8-bit A/Ds, one for monochrome data and one for chroma data; data derived to YCrCb

Pixel Jitter: ±6 nsec max

Aspect Ratio: 1:1 square pixels, depending on scaling factors

Frame Grab Speed: 1/30 s (60 Hz) or 1/25 s (50 Hz)

Modes: Interlaced (start on next even, next odd, or next field), single frame or continuous operation; all software selectable.

On-Board Processing

Region Of Interest: Programmable ROI window defines video data to be transferred to memory; pixels outside window are discarded

Scaling: Images scalable to 4 pixels by 4 lines, performed using linear phase interpolation; software selectable.

Data Formats

Image data can be output in 32-bit RGB, 15-Bit RGB and 16-bit YUV formats

Control Signals

External Trigger Inputs: One, TTL levels, software selectable on rising/falling edge

Sync Select: Sync can be generated internally or stripped from the video input channel used

Sync/Control Outputs: VSYNC, HSYNC, Composite Sync; signals provided to camera(s), board acts as Sync Master

Digital Inputs/Outputs: Four general-purpose bi-directional TTL input/output lines; TTL levels, fan-out of seven TTL loads each

Video Display

Uses PC's graphics card and monitor for display. Real-time video display and non-destructive, real-time animated overlays performed using Direct Draw (DDI)

Video Transfer Rate

20 MB/s typical, 132 MB/s max. Board operates as a Bus Master using Burst Mode for data transfer to host memory

Power Requirements

+5V @ 2 A typical

Physical/Environmental

Form: Half-size PCI bus board (short card)

Dimensions: 10.7cm x 17.5 cm (4.2 in. x 6.875 in.)

Weight: 150 g (5.3 ounces)

Operating Temperature: 0 °to 50 °C (32 °to 122 °F)

Storage Temperature: –25 °to 70 °C (–13 °to 158 °F)

Relative Humidity: Up to 90%, non-condensing

Warranty

One year limited parts and labor

System Requirements

Due to the high data-throughput available from the DT3153, the following PC configuration requirements are recommended:

- Pentium-class processor, 133 MHz or faster; Pentium II recommended
- At least one available PCI Bus slot
- Microsoft Windows 95, 98, or NT 4.0
- Triton PCI chipset (or better) and supporting system BIOS
- 16 MB of system RAM minimum for Windows 95/98; 32 MB minimum for Windows NT 4.0
- CD-ROM drive
- 3.5 in. high-capacity disk drive (for software installation)
- DDI-compatible graphics adapter

Ordering Summary

All Data Translation products are covered by a 1-year warranty. For prices please consult a price list, visit our web site, or contact your local reseller.

DT3153

The DT3153 is shipped with DT-Acquire™ software, and a Getting Started manual.

- DT3153

Note: Call for information on OEM and volume discounts.

Accessories

- EP306—1.5 m (5 ft.) cable assembly; accommodates three composite video inputs or one S-video and two composite inputs; external trigger input; and all four bi-directional digital I/O channels; connects to board using a mini D-shell connector

Software

All software packages include a copy of the software on CD-ROM, a user's manual, and 90 days of complimentary telephone support.

- DT-Active Open Layers
  ActiveX control for Microsoft Visual Basic 5.0 or higher, Visual C++ or higher, running under Windows 95/98 or Windows NT 4.0
  SP0974-CD
- 32-Bit Frame Grabber SDK for Windows 95/98 or Windows NT 4.0, Programmer's software compatible with Microsoft Visual C.
  SP0585-CL

For other compatible software, consult the Imaging section in this handbook.