Laboratório VISGRAF
Instituto de Matemática Pura e Aplicada

Um Sub-Sistema de Visão Computacional para Acompanahmento de Objetos

Bruno Madeira
Luiz Velho

Technical Report TR-02-01 Relatório Técnico

January - 2002 - Janeiro

The contents of this report are the sole responsibility of the authors.
O conteúdo do presente relatório é de única responsabilidade dos autores.

UM SUB-SISTEMA DE VISÃO COMPUTACIONAL PARA ACOMPAHAMENTO DE OBJETOS

POR

BRUNO EDUARDO MADEIRA

ORIENTAÇÃO

LUIZ VELHO

NOVEMBRO - 2001

Capítulo 1 - INTRODUÇÃO

A Visão Computacional Ativa tem por objetivo fundamental a simplificação dos problemas de Visão Computacional. Essa simplificação surge devido à restrições feitas no processo de aquisição das imagens. Mas especificamente, sistemas de Visão Computacional Ativa são capazes de posicionar seus sensores ópticos de forma a realizar uma melhor captação de informação.

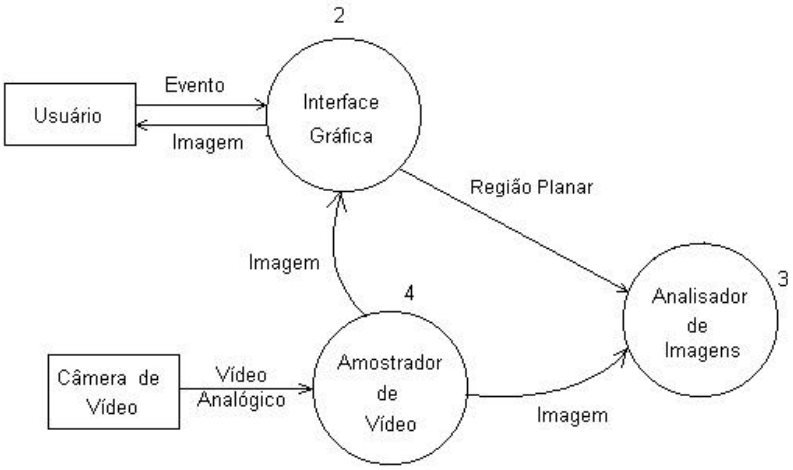
Este relátorio técnico descreve um sub-sistema de acompanhamento de objetos para visão computacional ativa. O sub-sistema faz parte de um sistema completo para sensoriamento guiado integrado por hardware e software específicos. O hardware consiste de uma câmera móvel acionada por motores e o software é um programa de visão computacional que controla a câmera [MADEIRA, 2001].

O sub-sistema de acompanhamento de objetos determina o movimento desejado da câmera, para que o objeto de interesse se mantenha sempre no centor da imagem corrente. Isso é feito baseado numa análise da sequência temporal de imagens. Por suas características, o sub-sistema de acompanahmento pode ser considerado o núcleo principal do sistema de sensoriamento guiado.

Capítulo 2 - ARQUITETURA DO SUB-SISTEMA

Esse capítulo descreve os componentes do sub-sistema de acompanhamento visual.

DFD-0



Capítulo 3 - FUNCIONALIDADE DOS MACRO-PROCESSOS

Será feita agora uma descrição em alto nível da funcionalidade fornecida por cada macro-processo apresentado no DFD-0 da sessão anterior:

3.1 - **Interface Gráfica**:

Fornece uma ferramenta interativa para a especificação de uma região a ser acompanhada no vídeo capturado. Permite a visualização em tempo real de um vídeo feito através da composição do vídeo capturado com uma curva delimitadora da região especificada.

3.2- **Analisador de Imagens**

Resolve o problema de correspondência das regiões selecionadas pela **Interface Gráfica** em amostras de vídeo consecutivas fornecidas pelo **Amostrador de Vídeo**.

3.3- **Amostrador de Vídeo**

Fornece um mecanismo de amostragem de vídeo.

CAPÍTULO 4 - AMOSTRADOR DE VÍDEO

4.1 – INTRODUÇÃO

Esse capítulo trata sobre o macro-processo **Amostrador de Vídeo.**

4.2 – *HARDWARE* UTILIZADO

Placa de digitalização de vídeo ***Captivator*.** Podendo ser substituída por qualquer outra baseada no chip Bt848 sem que seja necessária nenhuma modificação no *software.*

Pode ser empregado qualquer outro tipo de captura que suporte um *device driver* V4LINUX. Nesse caso o *driver* Bttv deverá ser trocado pelo *driver* V4LINUX específico do dispositivo empregado.

4.3 – *SOFTWARES* UTILIZADOS

Para a construção desse macro-processo foram utilizados os seguintes *softwares:*

- *GNU – **MPEG Library**, Gregory P. Ward.*
- *GNU – **Libbgrag**, A Shiffler*
- *GNU – **Bttv** V4LINUX device driver*
- ***SG3D*** (IMPA) - *Luiz Velho & Jonas Gomes*

4.4 – CARACTERÍSTICAS GERAIS

Os recursos oferecidos pelos *softwares* e pelo *hardware* descritos acima formam reunidos em uma interface comum capaz de oferecer serviços de amostragem de vídeos, que podem ser provenientes das seguintes origens:

1. Vídeo digitalizado em tempo real P&B ou Colorido
2. Vídeo armazenado em arquivo no formato MPEG
3. Vídeo sintetizado a partir de um *script* de animação tridimensional

Em sistemas de Visão Computacional Ativa a fonte de origem do sinal de vídeo é a que está descrita no primeiro item, ou seja, vídeo digitalizado em tempo real. As outras

possibilidades, entretanto são muito interessantes e úteis para o desenvolvimento de sistema de Visão Ativa.

Vídeos provenientes de arquivos MPEGs foram utilizados por exemplo para fins de teste dos macro-processo **Analisador de Imagens** e **Interface Gráfica**, atuando em vídeos que seriam muito difíceis de serem obtidos em tempo real como helicópteros, carros de combate, aviões...Etc.

Vídeos provenientes de *script* de animações tridimensionais são úteis por permitirem a realização de testes em situações difíceis de serem produzidas no mundo real. Esse tipo de operação não foi muito explorado devido à baixa qualidade do interpretador de *scripts* de animações que foi construído utilizando-se a biblioteca **SG3D**. O interpretador que foi construído não permite operações de mapeamento de textura em superfícies o que dificultou o trabalho do **Analisador de Imagens**, que precisava realizar o acompanhamento de objetos perfeitamente lisos.

CAPÍTULO 5 - INTERFACE GRÁFICA

5.1 – INTRODUÇÃO

Esse capítulo trata sobre o macro-processo **Interface Gráfica.**

5.2 – *SOFTWARES* UTILIZADOS

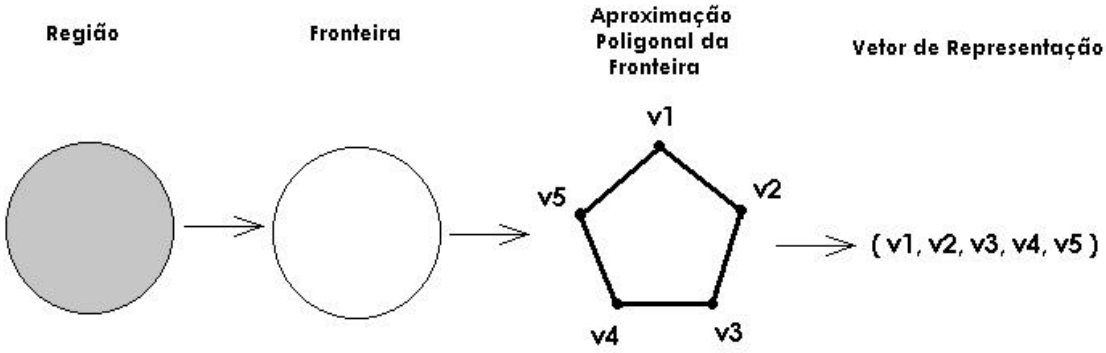
Para a construção desse macro-processo foram utilizados os seguintes *softwares:*

- ***OpenGL***
- ***GP*** (P.U.C) L. H. Figueiredo

5.3 – CARACTERÍSTICAS GERAIS

A função da **Interface Gráfica** é fornecer uma ferramenta interativa para a especificação de uma região a ser acompanhada no vídeo capturado.

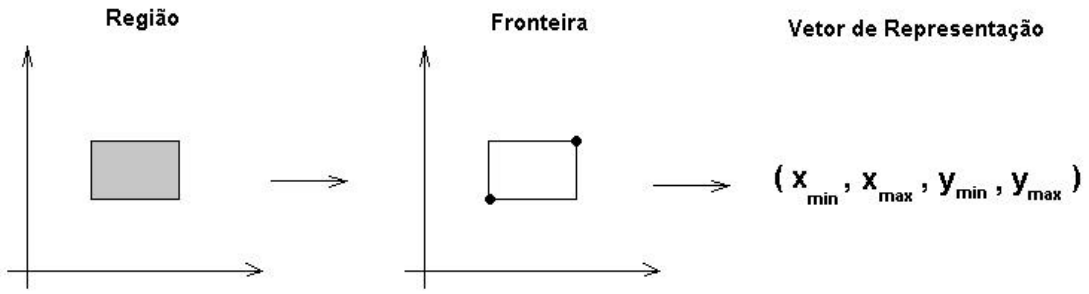
O método mais simples de especificar uma região é através das curvas que delimitam sua fronteira. Esse modelo matemático reduz o problema de representação de uma região para o problema de representação de curvas. A escolha da representação das fronteiras varia de acordo com a aplicação. Um exemplo de representação para curvas é a utilização de uma aproximação linear por partes, onde os vértices do polígono são armazenados em um vetor:



Em aplicações militares, o tempo que o usuário dispões para especificar a região é muito reduzido, além disso, o usuário muitas vezes encontra-se em condições adversas que prejudicam sua capacidade de raciocínio e sua coordenação motora. Essas peculiaridades das aplicações militares requerem a utilização de uma ferramenta de

especificação simples. Para construir ferramentas que satisfaçam esses requisitos é necessária a utilização de representações de poucos parâmetros.

No protótipo implementado, as regiões de interesse foram restringidas a regiões retangulares cujos lados são paralelos aos eixos coordenados. Esse esquema fornece uma representação da região como um vetor em R^4 como está ilustrado na figura abaixo:



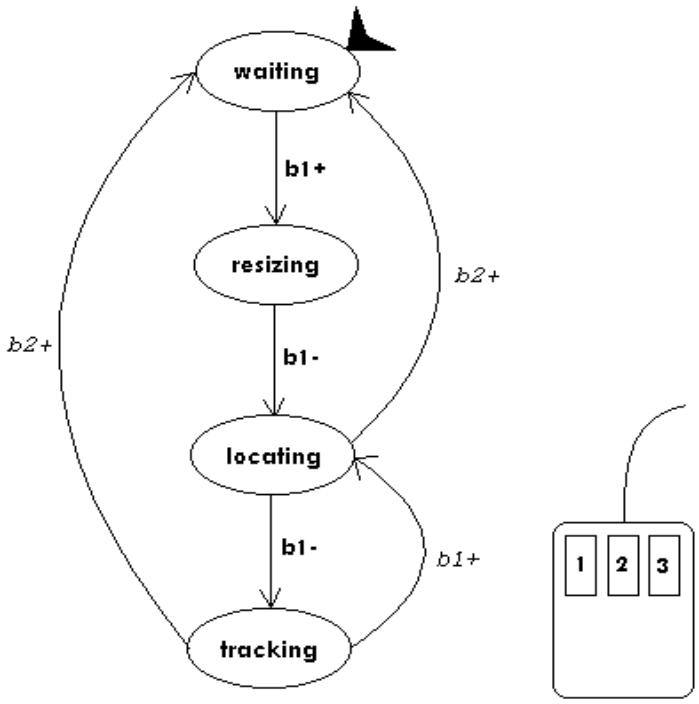
Essa região pode ser especificada de diversas maneiras. A cada maneira existe um conjunto de ações que devem ser executadas pelo usuário respeitando as especificações da máquina de estado definida pela interface gráfica.

A manipulação de eventos é feita por uma função que trabalha segundo uma máquina de estado cujas transições são determinadas pelos eventos que são passados como parâmetros para a função. Esses eventos são codificados por uma *string* e por um par ordenado (x, y), essa forma de codificação é estabelecida pelo ***GP.*** A codificação é descrita abaixo:

bi+	Botão i foi apertado;
bi-	Botão i foi liberado;
kt+	tecla t foi apertada;
ii+	cursor parado com botão i pressionado;
mi+	cursor em movimento com botão i apertado;
q+	janela foi fechada pelo gerenciador;
r+	requisição de redesenho.

Com exceção dos últimos dois eventos, o par ordenado (x, y) contém a posição do cursor no momento em que o evento ocorre.

Foi empregado um esquema de implementação onde é simples realizar a substituição da máquina de estados. Uma das possíveis máquinas de estados que o sistema pode utilizar está apresentada abaixo:



Obs:

Os eventos **b1+** e **b1-** correspondem a eventos do GP que podem ocorrer independente da posição do cursor. Já os eventos *b1+* e *b2+* devem ocorrer com o cursor posicionado sobre o retângulo definido no estado "resizing"

DESCRIÇÃO DOS ESTADOS

waiting: É o estado onde o sistema espera pela ação do usuário.

resizing: Enquanto o usuário mantiver o botão 1 pressionado, ele pode definir as dimensões da região alvo, que é o retângulo que será utilizado para envolver o objeto a ser acompanhado. Nesse estado o usuário define apenas as dimensões do retângulo movimentando o mouse. Para sair desse estado o usuário solta o botão 1. Esse estado é muito semelhante a operação de seleção da região a ser movimentada no *Paint Brush*.

locating: Para entrar nesse estado o usuário deve pressionar o botão 1 sobre o retângulo definida no estado ***resizing*.** Enquanto o botão 1 estiver pressionado, o usuário poderá "arrastar o retângulo" até que ele envolva a região alvo desejada. Esse estado é semelhante à operação de posicionamento da região de colagem no *Paint Brush*.

tracking: Nesse estado o sistema passa a acompanhar a região definida no estado ***locating*.** Pode-se sair desse estado de duas maneiras, ou apertando o botão 2 sobre a região alvo, o que apaga a região alvo e leva o sistema para o estado inicial; ou pressionando o botão 1 sobre o retângulo, o que permite posicionar a região alvo

Foram construídas rotinas para mapear os eventos do ***OpenGL*** definidos pela ***Glut*** para a codificação de eventos oferecida pelo ***GP***, isso permite a fácil utilização das mesmas máquinas de estados por eventos provenientes do ***OpenGL.***

5.4 – VANTAGENS E DESVANTAGENS ENTRE O OpenGL E O GP

5.4.1 – OpenGL

Vantagens:

- Possibilidade de uma aceleração por hardware;
- Pacote disponível para diversas plataformas diferentes;
- Padrão reconhecido mundialmente;

- Arquitetura que permite a visualização de imagens com mapeamento direto sobre a memória de vídeo, o que é muito eficiente.

Desvantagens:

- Dificuldades para migrar o ***OpenGL*** para uma tecnologia nova ou restrita;
- Embora o ***OpenGL*** permita que se acesse a memória diretamente, esse tipo de operação força a utilização do hardware em uma configuração específica, além disso, esse tipo de operação impossibilita mudança de escalas.

5.4.2 – GP

Vantagens:

- *API* muito simples
- Código fonte aberto
- Simplicidade na migração para uma tecnologia nova ou restrita.
- Pacote disponível para diversas plataformas diferentes

Desvantagens:

- Segue um paradigma meramente vetorial, por isso não possui o conceito de *pixel*.Quando se deseja visualizar uma imagem é necessário desenhar um retângulo para cada *pixel* da imagem que é uma tarefa ineficiente.

CAPÍTULO 6 - ANALISADOR DE IMAGENS

6.1 – INTRODUÇÃO

Esse capítulo trata sobre o macro-processo **Analisador de Imagens.**

6.2 – *SOFTWARES* UTILIZADOS

Para a implementação desse macro-processo, foi utilizado o *software* que utiliza o algoritmo ***KLT*** *(Kanade, Lucas, Tomasi)* implementado por *Stan Birchfield* em 1994.

6.3 – CARACTERÍSTICAS GERAIS

A função desse macro-processo é realizar a correspondência entre regiões em amostras consecutivas de um vídeo.

Foi utilizado o algoritmo ***KLT*** para a implementação desse macro-processo. Esse algoritmo, entretanto não foi utilizado em sua forma original tendo sido aplicado a ele uma heurística capaz de adapta-lo melhor a resolução do problema de correspondência entre regiões.

O algoritmo ***KLT*** pertence à classe dos algoritmos denominados *feature trackers.* Essa classe de algoritmos fornece a capacidade de realizar a correspondência entre regiões muito pequenas que possuam uma textura bastante marcante.

O algoritmo ***KLT*** foi escolhido por ser o primeiro algoritmo a fornecer a possibilidade de ser realizada a seleção de *features* ótimos para a realização do processo de correspondência. Antes desse algoritmo os *features* eram escolhidos mediante uma heurística de seleção onde se buscavam pontos onde o valor do módulo do gradiente da imagem era alto.

O ***KLT*** fornece então a solução para dois problemas de otimização:

1. O problema de seleção dos *features* ótimos em uma imagem
2. O problema de determinação da correspondência entre *features* em imagens consecutivas de um vídeo

6.4 – HEURÍSTICA UTILIZADA

Seria possível utilizar a solução do problema de correspondência do ***KLT*** em sua forma original para isso bastaria substituir o processo de seleção dos *features* ótimos da imagem por um processo de seleção feito pelo usuário através da **Interface Gráfica.** Essa solução, entretanto não conseguiria explorar o ponto forte do ***KLT*** que é a escolha de *features* ótimos. Por isso, decidiu-se aplicar uma heurística onde o ***KLT*** é bem empregado tornando a solução do problema de correspondência mais robusta. A heurística é descrita abaixo:

1. O usuário fornece a região a ser acompanhada através do posicionamento de um retângulo sobre a imagem.

2. O sistema passa ao *KLT* a sub-imagem delimitada pelo retângulo especificado, e solicita ao *KLT* que seja selecionado o *feature* cuja resolução do problema de correspondência é feita de maneira ótima. Esse *feature* é retornado com coordenadas medidas em um sistema de coordenadas intrínsecas ao retângulo.

3. O sistema determina a coordenada do *feature* selecionado no sistema de coordenadas da imagem.

4. O sistema solicita ao *KLT* que realize a correspondência entre esse *feature* e o *feature* na imagem consecutiva. O *feature* é substituído na imagem consecutiva pelo seu correspondente. Caso a correspondência não consiga ser estabelecida volta-se ao passo 2 utilizando o retângulo atual como parâmetro para o *KLT*.

5. Calcula-se um novo retângulo tal que as coordenadas do novo *feature* quando medidas no referencial intrínseco ao retângulo sejam iguais às coordenadas do *feature* na imagem anterior.

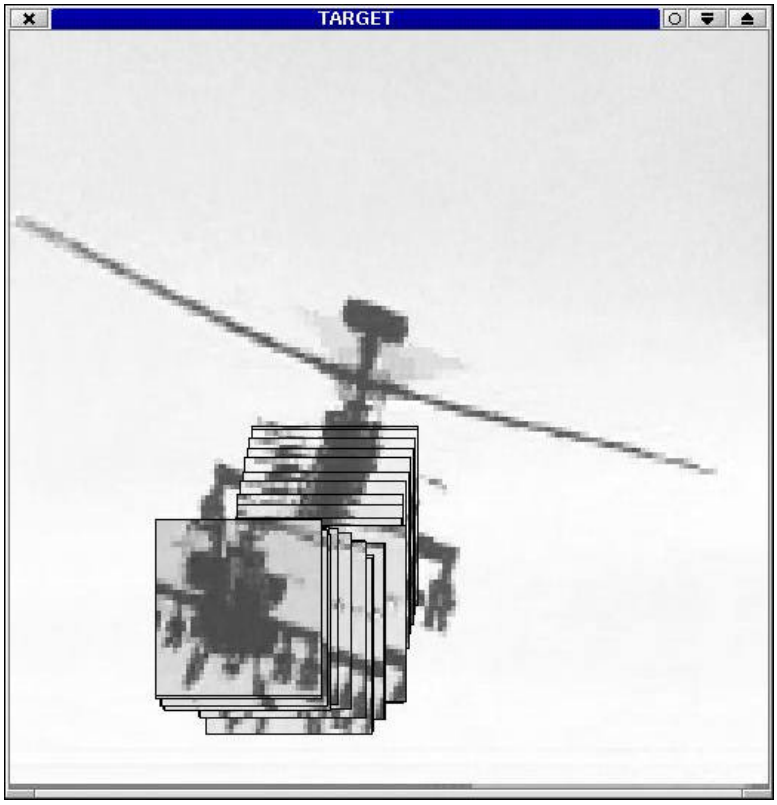
6. Retornar ao passo 4.

CAPÍTULO 7 - CONCLUSÕES

O protótipo implementado apresentou um resultado satisfatório. Abaixo são enumeradas algumas observações gerais e são dadas algumas sugestões para projetos futuros relacionados:

1) O algoritmo KLT mostrou-se adequado para o acompanhamento de uma classe de objetos em movimento bastante grande. Um bom trabalho posterior é a substituição do KLT por outros *feature trackers* para que seja feita uma comparação da eficiência desses algoritmos.
2) A heurística empregada no protótipo para a adequação do KLT mostrou-se adequada para diversas instâncias, entretanto falhou em alguns casos. Um trabalho que pode ser feito posteriormente é a substituição da heurística empregada atualmente por uma heurística que combine a informação proveniente de diversos *features*, gerando um acompanahmento mais robusto.
3) Não foi utilizado nenhum tipo de conhecimento a priori do tipo de instância ao qual o sistema de acompanhamento será submetido. A inserção de informações a priori pode tornar o sistema mais robusto.

Apêndice A - INTERFACE COM O USUÁRIO



REFERÊNCIAS BIBLIOGRÁFICAS

SHI, Jianbo, TOMASI, Carlo *Good Features to Track* IEEE Conference on Computer Vision and Pattern Recognition, pages 593-600, 1994

HORN, Berthold Kraus Paul *Robot Vision* The MIT Press McGraw-Hill Book Company

GOMES, Jonas, VELHO, Luiz *Computação Gráfica volume 1* Série de Computação e Matemática, IMPA. 1999

GOMES, Jonas, VELHO, Luiz "Sistemas Gráficos 3D" Série de Computação e Matemática, IMPA. 2001

WOO M., NEIDER J., DAVIS T., SHEREIDER D. *OpenGL Programming Guide* Addison-Wesley. 1999

MADEIRA, B., *Projeto e Implementação de Um Sistema de Vvisão Computacional Ativa.* Projeto Final de Graduação – IME. 2001