UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

RECONHECIMENTO AUTOMÁTICO DE LINHAS DE CAMPOS DE FUTEBOL EM ARQUIVOS DE VÍDEO PARA PUBLICIDADE VIRTUAL

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

EVERTON ELVIO KOSER

BLUMENAU, JUNHO/2003

2003/5-23

RECONHECIMENTO AUTOMÁTICO DE LINHAS DE CAMPOS DE FUTEBOL EM ARQUIVOS DE VÍDEO PARA PUBLICIDADE VIRTUAL

EVERTON ELVIO KOSER

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Paulo César Rodacki Gomes — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Paulo César Rodacki Gomes

_____

Prof. Jomi Fred Hübner

_____

Prof. Francisco Adell Péricas

RESUMO

Este trabalho apresenta um método para realizar o cálculo de calibração de câmeras, que representa a etapa inicial para implementação de um sistema de publicidade virtual. O objetivo de tais sistemas é a inserção dinâmica de imagens estáticas em cenas de vídeo contendo jogos de futebol. O método proposto utiliza como entrada de dados apenas arquivos de vídeo, e procura, através de visão computacional, inferir um sistema de coordenadas tridimensional para a cena. A validade da proposta é apresentada através da implementação de um protótipo em linguagem Java.

ABSTRACT

This work presents a method for camera calibration, which represents the initial stage for implementation of a virtual publicity system. The aim of such systems is the dynamic insertion of static images in video scenes of soccer games. The proposed method uses just video files as input data. Computational vision techniques are used in order to infer a three-dimensional coordinate system for the scene. The validity of the proposal is presented through the implementation of a prototype in Java language.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

No mundo dos esportes, pode-se perceber cada vez mais a utilização da computação gráfica. As transmissões de TV utilizam recursos como a inserção de logotipos ou painéis de seus patrocinadores nas suas imagens, jogadas decisivas são analisadas para determinar se um jogador estava impedido ou até mesmo ambientes virtuais são criados para dar ao telespectador uma visão privilegiada sobre outro ângulo do jogo.

A inserção dinâmica de logotipos em vídeos é definida por Gomes (1999) como Publicidade Virtual. Ela consiste em inserir imagens raster em um vídeo dando a impressão que elas realmente pareçam fazer parte do cenário do vídeo. Este trabalho propõe um método para realizar a Publicidade Virtual em vídeos contendo cenas de jogos de futebol.

Para realizar a Publicidade Virtual é necessário alinhar o sistema de coordenadas da câmera com o sistema de coordenadas do universo. Para alinhar os sistemas, deve-se aplicar diversas transformações geométricas a um dos sistemas para igualá-lo ao outro. Essas transformações são obtidas através do de algoritmos de calibragem de câmeras.

Presente trabalho apresenta os estágios iniciais de implementação de um sistema para publicidade virtual utilizando um método para realização de visão computacional com o objetivo de reconhecer as linhas do campo de futebol, para posteriormente inferir um sistema de coordenadas da câmera. O método abordado neste trabalho contém um algoritmo proposto por Szemberg (2001) para fazer o reconhecimento das linhas co campo. A estapas posteriores, não abordadas neste trabalho, prevêem a calibração da câmera que esteja filmando as cenas de jogos de futebol e a análise da projeção em perspectiva encontrada a partir das linhas do campo de futebol.

Para que a inserção virtual apresente um caráter mais realista, ela precisa ser deformada para contemplar a mesma visão em perspectiva gerada pela câmera. Para distorcer a imagem raster a ser inserida no vídeo, utiliza-se matrizes de transformações tridimensionais descritas por Mortenson (1999). Essas matrizes aplicam transformações geométricas como rotação, translação e escala à imagem, sendo assim possível manipulá-la conforme desejado. Conforme mencionado anteriormente, etapas posteriores à realização

do presente trabalho contemplam a implementação destas transformações, obtidas pela determinação da posição e perspectiva da câmera determinada pelo algoritmo de Szemberg (2001).

1.1 OBJETIVOS DO TRABALHO

Este trabalho tem como objetivo desenvolver as etapas iniciais de um método para realizar a Publicidade Virtual em arquivos de vídeo contendo jogos de futebol. Como entrada, será utilizado apenas o vídeo, as imagens que serão inseridas e as posições em que cada imagem deve ser inserida.

Este trabalho considera que os filmes utilizados como entrada sempre estejam filmando a região da grande área em um campo de futebol. Também considera que as câmeras tenham posições fixas no decorrer do vídeo.

1.2 ESTRUTURA DO TRABALHO

Primeiro será feita uma revisão bibliográfica dos diversos temas que são necessários para o desenvolvimento do método. Após isso é demonstrada a especificação do protótipo e alguns detalhes sobre a sua implementação. Finalmente serão apresentadas conclusões e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentadas considerações sobre os assuntos que serão tratados neste trabalho.

2.1 PUBLICIDADE VIRTUAL

Segundo Gomes (1999), Publicidade Virtual consiste basicamente na inserção eletrônica de anúncios publicitários dentro da imagem de vídeo, sendo que esta inserção seja feita de tal maneira que o tele-espectador tenha a impressão de que os anúncios fazem parte da cena realmente.

A Figura 1 mostra uma imagem raster simples, sem a aplicação de qualquer recurso de Publicidade Virtual. Já a Figura 2 apresenta a mesma imagem, porém com uma logomarca inserida dinamicamente utilizando recursos de Publicidade Virtual.

Figura 1 Imagem sem o uso da Publicidade Virtual

Figura 2 Imagem com o uso da Publicidade Virtual

Na Figura 2, pode-se perceber que o logotipo foi distorcido para se encaixar perfeitamente ao círculo central do gramado de um jogo de futebol. Se, ao invés de uma imagem estática, fosse utilizado um vídeo, o logotipo sempre seria distorcido para se encaixar exatamente nesta parte do campo, passando a impressão de que ele realmente faz parte do campo.

Outra característica da Publicidade Virtual, além do posicionamento geral da logomarca, está na maneira como ela é inserida. Os jogadores se encontram sobre a imagem, enquanto o campo encontra-se sob a mesma, reforçando ainda mais a impressão de que ela foi realmente pintada no gramado.

Para compreender o funcionamento da Publicidade Virtual, deve-se subdividi-la em 5 etapas distintas que serão tratadas separadamente no decorrer do capítulo. Abaixo a lista destas etapas:

a) pré-produção;
b) configuração;
c) operação ao vivo;
d) rendering;
e) resultado final.

2.1.1 PRÉ-PRODUÇÃO

Na pré-produção, é criada a imagem raster que será inserida no vídeo. Esse processo pode ser feito em qualquer editor gráfico e a imagem ainda não possui nenhuma relação com o vídeo em que ela será inserida.

2.1.2 CONFIGURAÇÃO

Na configuração do sistema, são realizadas as duas etapas mais delicadas do processo da Publicidade Virtual, a calibração da câmera e a marcação das áreas onde será inserida a imagem.

Detalhes sobre calibragem da câmera são encontrados na seção 2.2.2 Modelo de Câmera de Vídeo, 2.2.3 Calibragem da Câmera e em 2.3.4 Calibração de Câmeras. Por hora, basta saber que uma câmera está calibrada quando a sua posição com relação ao cenário é conhecida.

A marcação das áreas aonde será inserida a imagem consiste em demarcar no campo os locais aonde essas imagens serão inseridas.

2.1.3 OPERAÇÃO AO VIVO

Em uma aplicação normal da Publicidade Virtual, em que os dados vêm diretamente da câmera, esta etapa consiste em operar a câmera durante a filmagem do vídeo. Não apresentaria nenhuma diferença para o operador da câmera, ou seja, a filmagem será feita como se estivesse fazendo uma filmagem normal.

A única diferença é que a câmera retém informações importantes para as demais partes do processo, como o movimento de *pan* (movimento horizontal) e o *tilt* (movimento vertical).

Porém, como este trabalho aborta exclusivamente arquivos de vídeo como entrada, e não os dados provenientes da câmera, esses dados devem ser calculados nessa etapa do processo. Será utilizando o algoritmo proposto por Szemberg (2001) para fazer o cálculo

desses dados. Detalhes sobre o algoritmo de Szemberg (2001) são encontrados na seção 2.3 Algoritmo Para Calibragem de Câmeras.

Enquanto a pré-produção e a configuração acontecem com antecedência, antes da filmagem vídeo, a operação, o Rendering e o resultado final são executados ao vivo, ou seja, em tempo real.

2.1.4 RENDERING

O Rendering consiste em deformar e inserir a imagem raster no vídeo. O Apêndice A traz informações de como é feita a deformação da imagem raster para ser inserida no vídeo.

Para unir a imagem raster do vídeo existem várias técnicas, como será demonstrado a seguir.

2.1.4.1 TRANSPARÊNCIA

A transparência é utilizada quando se deseja ver através da imagem inserida, ou seja, deseja-se ver a imagem e o fundo ao mesmo tempo.

Figura 3 Uso da Transparência

A Figura 3 apresenta um exemplo da utilização da Transparência. Nesta figura, vários logotipos são inseridos através da Publicidade Virtual. Note que o logotipo da Rede Globo, sobre a piscina, está semitransparente, ou seja, é possível ver através dele, já os logotipos da Caixa Econômica Federal, no topo da arquibancada, não possuem qualquer transparência, sendo que não é possível ver nada através dele.

2.1.4.2 *CHROMA KEY*

Segundo Gomes (1999) e Tonietto (2003), o *chroma key* é utilizado em inserções virtuais com recursos de oclusão. A técnica de *chroma key* permite que uma determinada cor seja isolada e que uma outra imagem (uma inserção virtual) seja "pintada" nos locais onde esta cor aparece.

Figura 4 Uso do *Chroma Key*

Por exemplo, na Figura 4, é inserida uma textura no centro de um campo de futebol, deve-se usar uma amostra da cor verde do piso e dizer para o sistema inserir a textura somente nas partes da região demarcada que contém esta cor verde. Portanto, se um jogador com uniforme amarelo correr no círculo central, sua silhueta esconderá o verde do piso e estes lugares do desenho não serão pintados na tela, dando a impressão de que o jogador está realmente correndo sobre um desenho pintado no piso. Analogamente, se um jogador

usando um uniforme de cor verde muito parecida com o verde do piso passar pelo círculo central, seu uniforme irá desaparecer por baixo do desenho inserido.

2.2 CÂMERAS DE VÍDEO

Nesta seção serão demonstradas algumas das propriedades das câmeras de vídeo, além de alguns modelos matemáticos de câmeras.

2.2.1 PROPRIEDADES DAS CÂMERAS DE VÍDEO

As câmeras de vídeo possuem várias propriedades ou características, que são de grande importância para a aplicação da Publicidade Virtual. Neste item serão abordadas as principais propriedades existentes das câmeras.

Segundo Gomes (1999) e Szemberg (2001), o processo de determinar as propriedades de câmeras de vídeo, denomina-se calibração de câmeras, e uma câmera só está calibrada a partir do momento que suas propriedades forem conhecidas.

2.2.1.1 POSICIONAMENTO RELATIVO A CENA

O posicionamento da câmera com relação à cena é a determinação das coordenadas X, Y e Z em um sistema de coordenadas que a câmera ocupa com relação à cena. Não é propriamente uma propriedade de câmera, porém a sua determinação é de fundamental importância para a Publicidade Virtual. Neste trabalho está sendo considerado que as câmeras tenham posições fixas no decorrer do vídeo.

2.2.1.2 MOVIMENTAÇÃO DA CÂMERA

A movimentação da câmera é composta pelos ângulos horizontal e vertical que a câmera possui com relação à cena. Eles são chamados de Pan e Tilt, respectivamente.

2.2.1.3 ZOOM

O Zoom determina a ampliação da imagem na cena. Quanto maior o Zoom, maior será o tamanho dos objetos presentes no vídeo.

2.2.1.4 CENTRO ÓPTICO DA LENTE

O centro óptico é um ponto imaginário na lente que serve como base para a determinação dos demais parâmetros da lente. Ele dificilmente coincide com seu centro físico (geométrico). É de fundamental importância saber corretamente a posição do centro óptico, pois todas as outras informações para a operação correta e eficiente do sistema dependem disso.

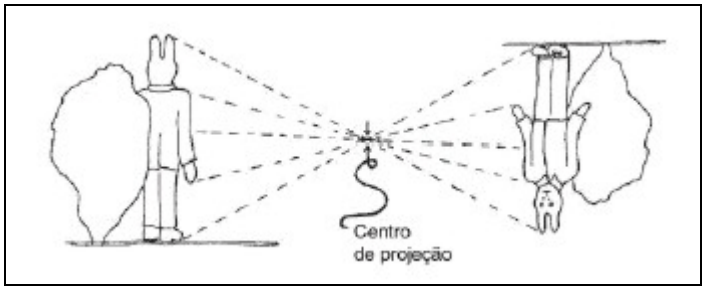
2.2.1.5 ABERRAÇÕES

Aberrações são distorções que as imagens do vídeo apresentam ao se afastar do centro óptico da lente. As aberrações representam um grande obstáculo para ser contornado, pois ela transforma linhas retas em curvas, as linhas que são paralelas no mundo real não são mais paralelas no vídeo e vice versa. Em imagens de vídeo, isto pode ser verificado na Figura 4, observando-se como o logotipo da Skol, no canto superior direito está distorcido.

2.2.2 MODELO DE CÂMERA DE VÍDEO

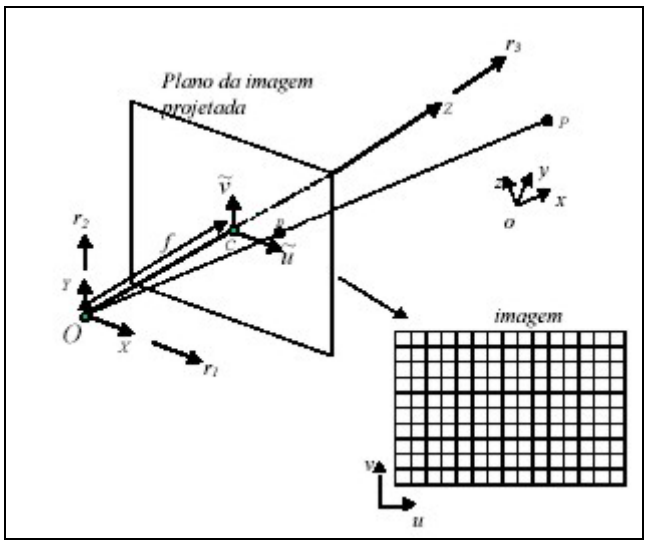
Neste trabalho será utilizado o modelo de câmera *Pinhole*, que pode ser visto em Szemberg (2001). Este modelo visa capturar as imagens do mundo real que passam por um orifício e são projetadas em um plano. A Figura 5 demonstra como funciona a câmera de *Pinhole*.

Figura 5 Projeção através de um ponto



Em computação gráfica, esse modelo é simplificado ainda mais, como demonstra a Figura 6. Este é o modelo que será utilizado mais adiante neste trabalho para calibrar a câmera.

Figura 6 Modelo de câmera Pinhole simplificado



O ponto *O* representa o centro de projeção da câmera, é neste ponto que todos os raios de luz vindos da cena convergem. O ponto *C* representa o centro geográfico do plano que as imagens são capturadas, este plano, cujo ponto C é o centro, dá origem às imagens de vídeo correspondendo portanto à área de visualização na tela do monitor. Os pontos neste plano são mapeados por um sistema de coordenadas que possue apenas duas dimensões, *u* e *v*. A distância entre o ponto *O* e o ponto *C*, mostrada com o símbolo f, interfere diretamente sobre o zoom das imagens produzidas.

O ponto *o* representa a origem de um sistema de coordenadas lógico, independente da câmera. Neste trabalho, o modelo lógico deve ser um campo de futebol com um tamanho fixo. Em um sistema de publicidade virtual, os logotipos devem ser inseridos neste modelo lógico, facilitando assim o trabalho do usuário em inserir os logos. Tendo a câmera calibrada, é possível mapear o sistema de coordenadas o(x,y,z) para o sistema O(X,Y,Z) e utilizá-lo com base na visualização oferecida pela câmera.

2.2.3 CALIBRAGEM DA CÂMERA

O processo de calibrar a câmera consiste em uma maneira de transformar um sistema de coordenadas lógico no próprio sistema de coordenadas da câmera. Este processo é muito importante não apenas para a Publicidade Virtual, mais também para muitas outras aplicações na Computação Gráfica. Muitos efeitos dinâmicos como a inserção de imagens, cálculo da velocidade da bola, verificação de impedimentos, além mitos outros não citados aqui dependem de uma boa calibragem da câmera.

O Quadro 1 demonstra a equação utilizada para gerar as coordenadas $\tilde{u}$ e $\tilde{v}$ utilizando o sistema de coordenadas próprio da câmera, $C(X,Y,Z)$.

Quadro 1 Equação de calibração utilizando o sistema da câmera

$$\boxed{(\tilde{u},\tilde{v}) = \left(f\frac{X}{Z}, f\frac{Y}{Z} \right)}$$

Para poder gerar as coordenadas $\tilde{u}$ e $\tilde{v}$ de um sistema de coordenadas lógico, deve-se mapear esse sistema, sendo necessário uma translação e em seguida uma rotação para se alinhar os dois sistemas. Detalhes sobre como são feitas as transformações geométricas dos objetos estão no Apêndice A.

As translações e rotações individuais são desprezíveis para a conversão dos sistemas, o que realmente importa é a união de todas as transformações. O Quadro 2 demonstra a equação genérica para a conversão entre os dois sistemas de coordenadas. A matriz 3x4 contém a união de todas das transformações geométricas sofridas pelo sistema, que são representadas individualmente pelos símbolos q_{11} até q_{34}. A obtenção de seus valores será vista mais adiante na seção 2.3.4 Calibração de Câmeras.

Quadro 2 Conversão de Sistemas

$$\boxed{o\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} * \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \end{bmatrix} = C\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}$$

Então aplicando as fórmulas presentes nos Quadro 1 e Quadro 2, já é possível mapear as coordenadas de $\tilde{u}$ e $\tilde{v}$ de qualquer sistema de coordenadas, assim como o seu processo inverso, determinar as coordenadas x, y e z dos pontos partindo de $\tilde{u}$ e $\tilde{v}$. As equações presentes nos Quadros 3 e 4 demonstram como é feita essa conversão.

Quadro 3 Coordenada $\tilde{u}$ para um sistema genérico

$$\tilde{u} = \frac{(q_{11} * x) + (q_{12} * y) + (q_{13} * z) + (q_{14} * 1)}{(q_{31} * x) + (q_{32} * y) + (q_{33} * z) + (q_{34} * 1)}$$

Quadro 4 Coordenada $\tilde{v}$ para um sistema genérico

$$\tilde{v} = \frac{(q_{11} * x) + (q_{12} * y) + (q_{13} * z) + (q_{14} * 1)}{(q_{31} * x) + (q_{32} * y) + (q_{33} * z) + (q_{34} * 1)}$$

2.3 ALGORITMO PARA CALIBRAGEM DE CÂMERAS

Szemberg (2001) criou um algoritmo para calibragem de câmeras mostrando cenas de jogos de futebol. Este algoritmo se encaixa perfeitamente as necessidades deste trabalho, pois ele calibra uma câmera utilizando como entrada apenas o vídeo contendo os jogos de futebol, que é a principal dificuldade encontrada neste trabalho. O seu funcionamento básico consiste em achar as linhas do campo, e baseado na perspectiva entre elas, calcular as diversas propriedades da câmera.

O algoritmo de Szemberg (2001) é, na verdade, um conjunto de outros algoritmos sobre Computação Gráfica e Percepção Visual. Para o explicar mais detalhadamente, será feita uma abordagem por camadas, ou seja, será feita uma divisão lógica de cada etapa do processo, e internamente, será feita uma sub-divisão que descreve separadamente os algoritmos que compõem a parte lógica.

2.3.1 FILTRAGEM E SEGMENTAÇÃO

Esta etapa do processo consiste em detectar pontos passíveis de estarem sobre linhas presentes na imagem e suavizar problemas presentes na imagem, como ruídos.

A Figura 7 e a Figura 11 demonstram a entrada e saída, respectivamente dessa etapa do processo.

Figura 7 Imagem de entrada do Algoritmo

2.3.1.1 IMAGENS DE VÍDEO

Esta parte do algoritmo tem a função transformar a imagem colorida da entrada em outra monocromática e aplicar um filtro negativo a essa imagem.

Para poder falar sobre como a imagem de entrada será transformada em uma imagem monocromática, é necessário falar sobre o formato do vídeo de entrada. Cada frame do vídeo é composto por um vetor de bytes, que tem a função de mostrar as cores de cada pixel da imagem do vídeo, segundo a escala de cor RGB.

Segundo Szemberg (2001) e Bertulani (2000), canal de cor RGB é uma escala para demonstrar as cores. Ele é formado pela decomposição das três cores primárias, vermelho, verde e azul, do inglês *red*, *green* e *blue*, daí o nome RGB. Cada cor primária é composta por 1 byte, sendo que o 0 indica a ausência total de cor e o 255 a presença máxima da mesma. Como existem 3 cores primárias, cada pixel descrito pelo canal de cor RGB ocupa 3 bytes.

São utilizados dois filtros para transformar a imagem colorida em monocromática. O primeiro filtro, é o descarte de canais de cor, em que a imagem monocromática é formada pegando apenas um canal de cor RGB da imagem colorida e descartando os outros dois canais. Este método é mais rápido, porém, informações importantes podem ser perdidas, pois dois canais de cor são descartados. O segundo filtro, é a Luminância, em que a

imagem monocromática é formada aplicando um peso a cada canal de cor RGB da imagem de origem, conforme o Quadro 5.

Quadro 5 Luminância

$$L = 0{,}299R + 0{,}587G + 0{,}114B$$

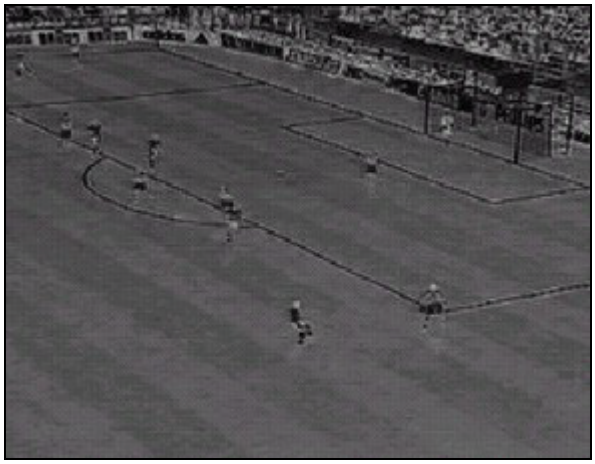
O filtro Negativo consiste em inverter as cores, ou seja, transformar o preto em branco e o branco em preto. Como existem apenas 256 tonalidades de cinza, a imagem, basta aplicar a equação do Quadro 6 à imagem monocromática da etapa anterior.

Quadro 6 Negativo

$$N = 255 - I$$

A Figura 8 demonstra o resultado da aplicação do filtro Luminância e do filtro Negativo à Figura 7.

Figura 8 Negativo da Imagem Mocromática



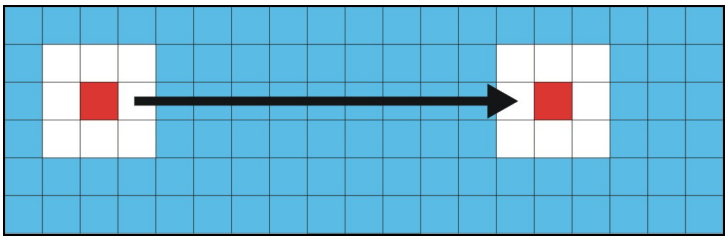
2.3.1.2 REALCE DE LINHAS

Esta etapa consiste em realçar as diferenças de tonalidade, presentes na figura, como forma de destacar os objetos presentes na mesma. Para isso, será utilizado um filtro conhecido como *LoG*, do inglês *Laplacian of Gaussian* ou Laplaciano do Gaussiano.

Esses filtros são diferentes do filtro de luminância, visto na secção 2.3.1.1, pois cada ponto da imagem criada depende de um conjunto de pontos da imagem original. Este tipo de filtro é chamado de Kernel. A figura 9**Erro! Fonte de referência não encontrada.**

demonstra a utilização do Kernel, os quadrados azuis representam a imagem de origem, os brancos a matriz do kernel e os vermelhos representam o resultado da aplicação do filtro. Para calcular o Kernel de um ponto, deve-se fazer o somatório do produto de cada ponto da imagem original sob a matriz pelo próprio valor da matriz sobre ele. Mais informações sobre ele são encontradas em HIPR2 (2000).

Figura 9 Kernel



Primeiro aplica-se o Kernel proposto por Gaussian (Quadro 7) visando minimizar os ruídos da imagem. Após isso, aplica-se o Kernel de Laplacian (Quadro 8) que identifica diferenças de tonalidade na figura.

Quadro 7 Kernel do filtro de Gaussian

1	2	1
2	4	2
1	2	1

Quadro 8 Kernel do filtro de Laplacian

0	1	0
1	-4	1
0	1	0

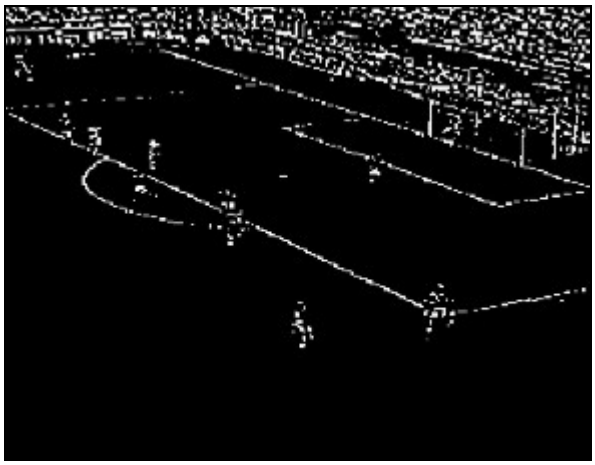
A Figura 10 demonstra a aplicação do filtro LoG a imagem do processo anterior.

Figura 10 Filtro de LoG

2.3.1.3 SEGMENTAÇÃO DA IMAGEM

Esta etapa consiste em filtrar os pontos que são aptos à fazer parte de um objeto qualquer, ele consiste em transformar a imagem resultante do filtro *LoG*, formada por tons de cinza, em uma outra imagem binária preto e branco. Se o valor de um pixel resultante do filtro *LoG* possuir um valor escalar acima de um número pré-definido, ele é selecionado, caso contrário, é descartado. A Figura 11 demonstra a utilização desse filtro.

Figura 11 Filtro de Segmentação por Limiar

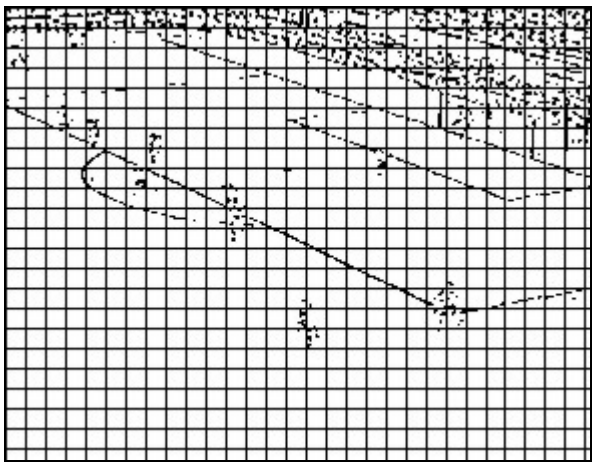


2.3.2 EXTRAÇÃO DOS SEGMENTOS DE RETA

Esta etapa do processo visa identificar e extrair todos os segmentos de reta existentes na imagem, independente de fazerem parte do campo ou não.

Todos os filtros encontrados nesta etapa do processo exigem que a imagem esteja subdividida em retângulos de iguais proporções, conforme a Figura 12.

Figura 12 Imagem quadriculada



2.3.2.1 ELIMINAÇÃO DE PONTOS

O objetivo dessa etapa do processo é eliminar os quadrados indesejados, que são os que não possuem pontos ou os que os pontos presentes não formem uma reta. Para fazer a eliminação das células, serão utilizados os autovalores λ_1 e λ_2, referentes à matriz de covariância calculada para cada célula.

Esta matriz é representada em Quadro 9. Os seus elementos são calculados em Quadro 10, Quadro 11 e Quadro 12. Deve-se considerar n como sendo o número de pontos na célula, u_i e v_i as coordenadas cartesianas de cada ponto de um retânculo e $\bar{u}$ e $\bar{v}$ são as coordenadas cartesianas do centróide dos pontos, que é obtido atravéz da média dos pontos u_i e v_i.

Quadro 9 Matriz de covariância

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

Quadro 10 Valor "a" da mariz de covariância

$$a = \frac{\sum_{i=1}^{n} \left(u_i - \bar{u}\right)^2}{n}$$

Quadro 11 Valor "b" da matriz de covariância

$$b = \frac{\sum_{i=1}^{n} \left(u_i - \bar{u}\right)\left(v_i - \bar{v}\right)}{n}$$

Quadro 12 Valor "c" da matriz de covariância

$$c = \frac{\sum_{i=1}^{n} \left(v_i - \bar{v}\right)^2}{n}$$

Quadro 13 Autovalor λ_1

$$\lambda_1 = \frac{a + c + \sqrt{(a-c)^2 + 4b^2}}{2}$$

Quadro 14 Autovalor λ_2

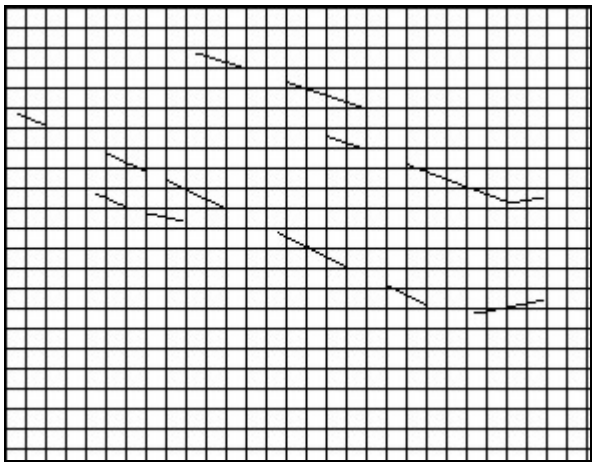
$$\lambda_2 = \frac{a + c - \sqrt{(a-c)^2 + 4b^2}}{2}$$

Os autovalores λ_1 e λ_2 são calculados nos Quadro 13 e em Quadro 14. O autovalor λ_1 representa a maior distância entre os pontos do quadrado, e o autovalor λ_2 representa a menor distância.

Quanto maior a razão entre eles, mais perfeita será a reta existente no quadrado. Então são descartados todos os quadrados que a razão entre os autovalores seja inferior a uma constante.

A Figura 13 demonstra a utilização dos autovalores tendo como origem a Figura 12.

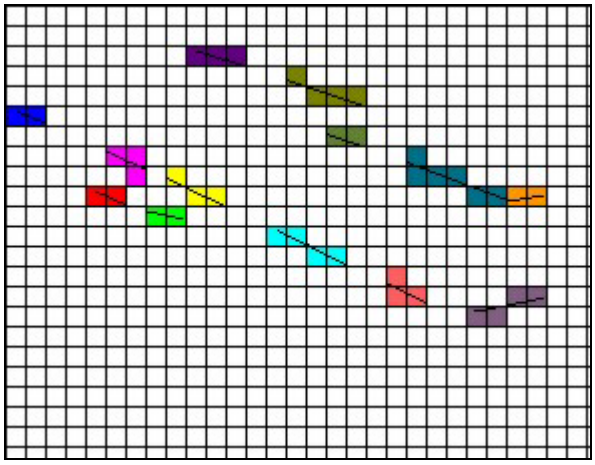
Figura 13 Resultado do método dos autovalores



2.3.2.2 ATRIBUIÇÃO DE VALORES AOS QUADRADOS

Esta etapa do processo consiste em verificar se a linha encontrada em um quadrado continua nos seus quadrados vizinhos. Para fazer isso, são utilizados autovalores com a união dos pontos do quadrado atual com os pontos de um de seus vizinhos. A Figura 14 demonstra o resultado deste procedimento.

Figura 14 Quadrados com valores atribuídos indicados pelas cores



2.3.2.3 EXTRAÇÃO DOS SEGMENTOS DE RETA

A extração dos segmentos de reta consiste em determinar a equação de reta de cada segmento encontrado na seção anterior. Para determinar a equação de reta é utilizado o método dos Mínimos Quadrados.

O método dos Mínimos Quadrados visa encontrar a equação de reta descrita no Quadro 15, sendo que u e v as coordenadas de cada ponto e a e b constantes, que são calculados em Quadro 16.

Quadro 15 Equação de uma Reta

$$v = au + b$$

Quadro 16 Equação do Mínimos Quadrados

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} u_i^{\,2} & \sum_{i=1}^{n} u_i \\ \sum_{i=1}^{n} u_i & n \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} u_i v_i \\ \sum_{i=1}^{n} v_i \end{bmatrix}$$

2.3.2.4 UNIÃO DOS SEGMENTOS DE RETA

Esta etapa consiste em unir as linhas do campo que possuem a mesma direção e estão separadas, pois existem quadrados entre elas sem nenhuma linha.

As linhas de um campo de futebol são ortogonais entre si, porém existem algumas variáveis que distorcem essas características, como por exemplo, as Aberrações, vistas em 2.2.1.5 Aberrações. Então é necessário encontrar os segmentos de reta que estejam sobre a mesma reta e que possuam uma diferença de ângulo pequena entre si.

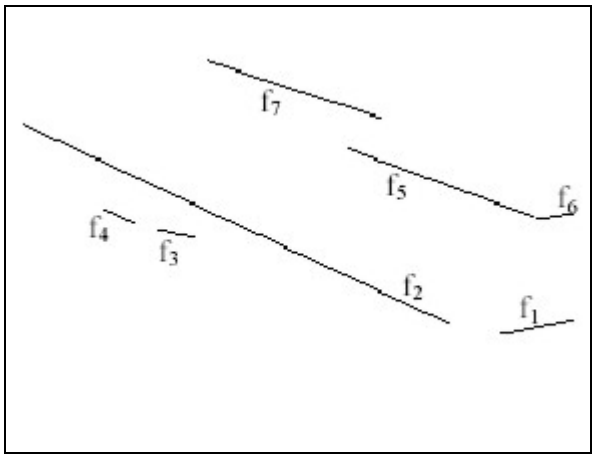
A Quadro 17 demonstra como calcular o ângulo entre dois segmentos de reta. Se α for menor que um dado valor, então os dois segmentos são partes de uma mesma reta. Deve-se criar uma nova reta a partir da união dos pontos dos dois segmentos.

Quadro 17 Diferença de angulo entre dois segmentos de reta

$$\alpha = \frac{\left|\overrightarrow{ab} * \overrightarrow{cd}\right|}{\left\|\overrightarrow{ab}\right| * \left|\overrightarrow{cd}\right\|}$$

A Figura 15 demonstra o resultado da união dos segmentos de reta.

Figura 15 União dos segmentos de Reta

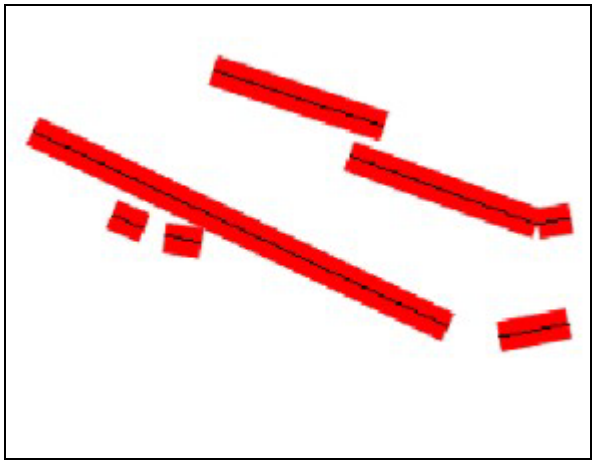


2.3.2.5 REAJUSTE DE LINHAS

Como todo o processo de detecção das linhas foi processado separadamente, com base nos quadrados formados a partir da Figura 12, muitos dados importantes com relação a imagem inteira foram perdidos. Para minimizar essa perda, deve-se reajustar as linhas.

Para fazer o re-ajuste das linhas, deve-se selecionar todos os pontos encontrados em uma área ao redor das linhas encontradas, conforme a Figura 13, essas áreas, que estão pintadas em vermelho, são denominadas faixas de tolerância. Uma vez tendo esses pontos encontrados, deve-se calcular a equação de reta novamente com base nos Mínimos Quadrados.

Figura 16 Reajuste das retas

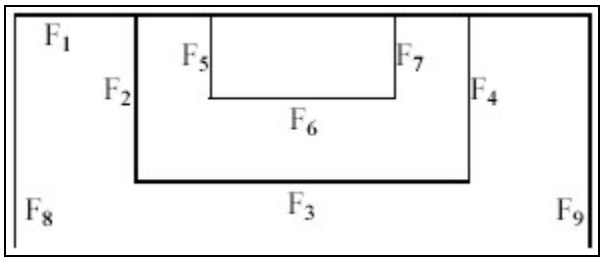


2.3.3 RECONHECIMENTO E INTERPRETAÇÃO

Nesta etapa do processo todas as linhas presentes na cena já foram identificadas. O próximo passo é interpretá-las para descobrir quais linhas fazem parte do campo. É justamente isso que esta seção se propõe a fazer.

Szemberg (2001) demonstra a técnica utilizada para fazer o reconhecimento das linhas, denominada Método de Reconhecimento Baseado em Modelos. Este método compara as linhas encontradas na etapa anterior, com as linhas existentes em um modelo geométrico pré-definido, visando identificar as linhas equivalentes. A Figura 17 mostra o modelo utilizado neste trabalho.

Figura 17 Modelo real de um campo de futebol



Antes de fazer esta comparação, algumas considerações devem ser feitas:

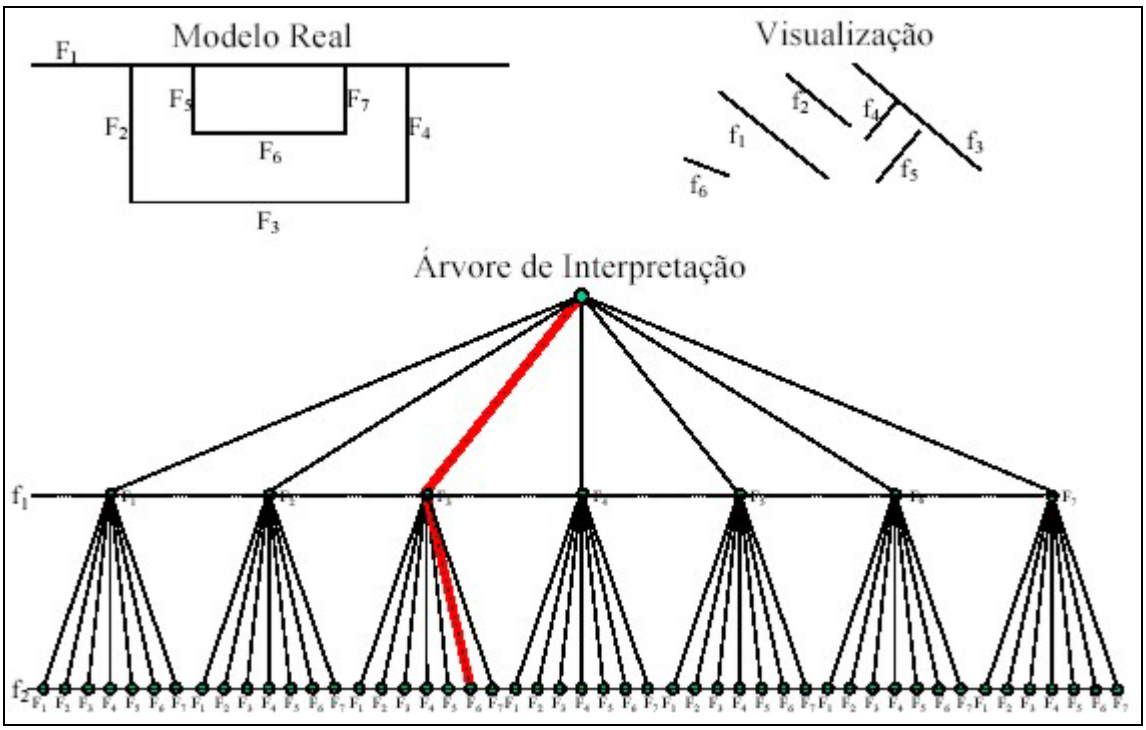
a) uma linha na visualização não necessariamente deve ter uma equivalente no modelo, já que nem todas as linhas encontradas anteriormente fazem parte do campo;
b) uma linha do modelo não necessariamente deve ter uma equivalente na visualização, pois algumas linhas não são encontradas nos processos anteriores, além de que, a área da cena filmada pode não cobrir o modelo inteiramente.

Tendo conhecimento dessas restrições, deve-se criar uma árvore que contenha todas as combinações possíveis entre as linhas do modelo e as linhas da visualização. Essa árvore é denominada árvore de interpretação. A Figura 18 demonstra os dois primeiros níveis de uma árvore de interpretação.

Na Figura 18, os nós F_i correspondem às i linhas do modelo real. Cada nível da árvore (indicado por f1 e f2 no lado esquerdo da mesma) correspondem a linhas calculadas

na visualização do vídeo. Na árvore, a linha calculada f_1 corresponde ao nó F_3 do modelo real, e a linha calculada f_2 da visualização corresponde ao nó da linha F_6 do modelo real.

Figura 18 Exemplo de uma Arvore de interpretação



Uma vez tendo a árvore de interpretação criada, deve-se descobrir qual de seus filhos contém a melhor solução, a solução na qual a visualização se encaixe com o modelo. Esta solução está marcada com a cor vermelha na Figura 18. Para descobrí-la, o modelo deve ser convertido em regras geométricas, conforme o Quadro 18.

A seguir, deve-se verificar quantas regras são válidas para cada filho da árvore de interpretação. O filho que atender o maior número de regras é a solução em que a visualização mais se encaixa com o modelo.

Quadro 18 Modelo convertido em Regras Geométricas

a) duas linhas na visualização não podem ter a mesma representação no modelo;
b) duas linhas são paralelas (ou próximas de paralelas, em virtude da transformação projetiva) na visualização se, e somente se, suas representantes no modelo real também forem paralelas;
c) todas as linhas devem estar em um mesmo semiplano determinado pela linha que representa F1;
d) todas as linhas devem estar em um mesmo semiplano determinado pela linha que representa F8;
e) todas as linhas devem estar em um mesmo semiplano determinado pela linha que representa F9;
f) todas as linhas devem estar em um mesmo semiplano determinado pela linha que representa F2, exceto as representantes de F1 e F8;
g) todas as linhas devem estar em um mesmo semiplano determinado pela linha que representa F3, exceto as representantes de F8 e F9;
h) todas as linhas devem estar em um mesmo semiplano determinado pela linha que representa F4, exceto as representantes de F1 e F9;
i) a linha que representa F5 deve estar entre as linhas que representam F1 e F6;
j) a linha que representa F5 deve estar entre as linhas que representam F2 e F6;
k) a linha que representa F6 deve estar entre as linhas que representam F1 e F3;
l) a linha que representa F7 deve estar entre as linhas que representam F1 e F6;
m) a linha que representa F7 deve estar entre as linhas que representam F4 e F6;
n) o correspondente de F5 não pode cruzar o correspondente de F6 e vice-versa;
o) correspondente de F7 não pode cruzar o correspondente de F6 e vice-versa.

2.3.4 CALIBRAÇÃO DE CÂMERAS

Esta seção, demonstra como utilizar os dados encontrados na seção anterior para calibrar uma câmera de *Pinhole*, vista na seção 2.2.2 Modelo de Câmera de Vídeo.

Para poder resolver as equações encontradas em Quadro 3 e Quadro 4, deve-se selecionar pelo menos 4 pontos de referência, que tenham uma posições determinadas tanto no sistema, $C(\tilde{u},\tilde{v})$ quanto no Sistema do Universo, U(x,y,z). Esses pontos são locais específicos no campo de futebol como a marca do pênalti, interseção das linhas da área e assim por diante.

Tendo esses 4 pontos selecionados temos dois sistemas com 4 variáveis e 4 equações diferentes, sendo assim matematicamente possível resolvê-lo.

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo serão discutidas as atividades relacionadas com a elaboração e desenvolvimento do protótipo proposto por este trabalho.

3.1 REQUISITOS DO PROBLEMA

Para realizar a Publicidade Virtual, deve-se passar para o sistema um arquivo de vídeo contendo o jogo de futebol, uma lista contendo as imagens raster que desejam ser inseridas e as matrizes de transformação de cada imagem.

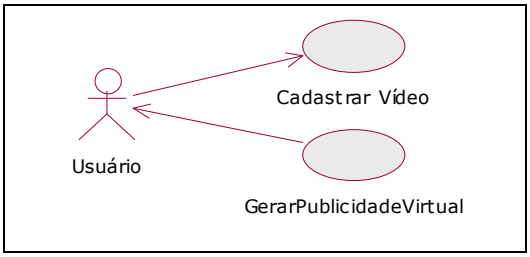
O sistema deve retornar um vídeo em que as imagens raster estejam inseridas dinamicamente no vídeo, conforme as regras da Publicidade Virtual.

3.2 ESPECIFICAÇÃO

Para especificar o problema será utilizado UML, descrito por Furlan (1998). Serão apresentados diagramas de casos de uso, de classes e seqüência do protótipo. Como os objetivos do trabalho não foram concluídos em sua totalidade, serão apresentados os diagramas apenas das partes que foram efetivamente implementadas.

3.2.1 DIAGRAMA DE CASOS DE USO

Figura 19 Diagrama de Casos de Uso

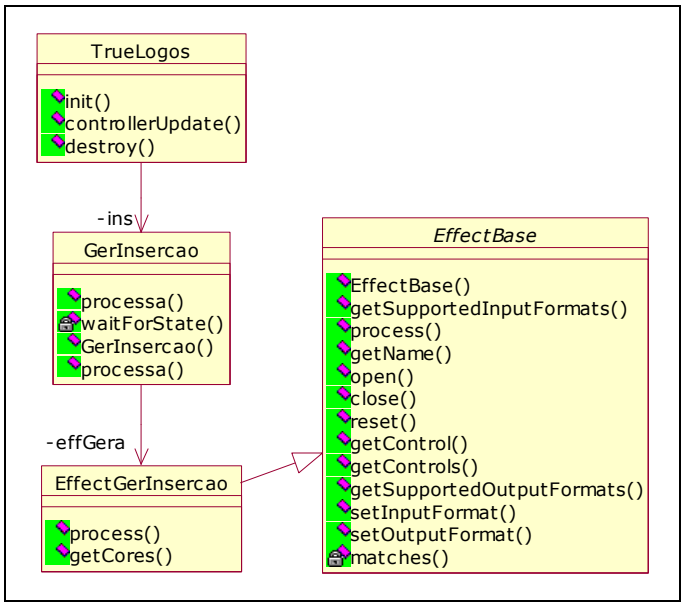


A Figura 19 demonstra o Diagrama de Casos de Uso do protótipo a ser desenvolvido. O Usuário deve cadastrar o vídeo contendo o jogo e o protótipo deve gerar como retorno para o usuário um vídeo com a publicidade virtual realizada.

3.2.2 DIAGRAMAS DE CLASSES

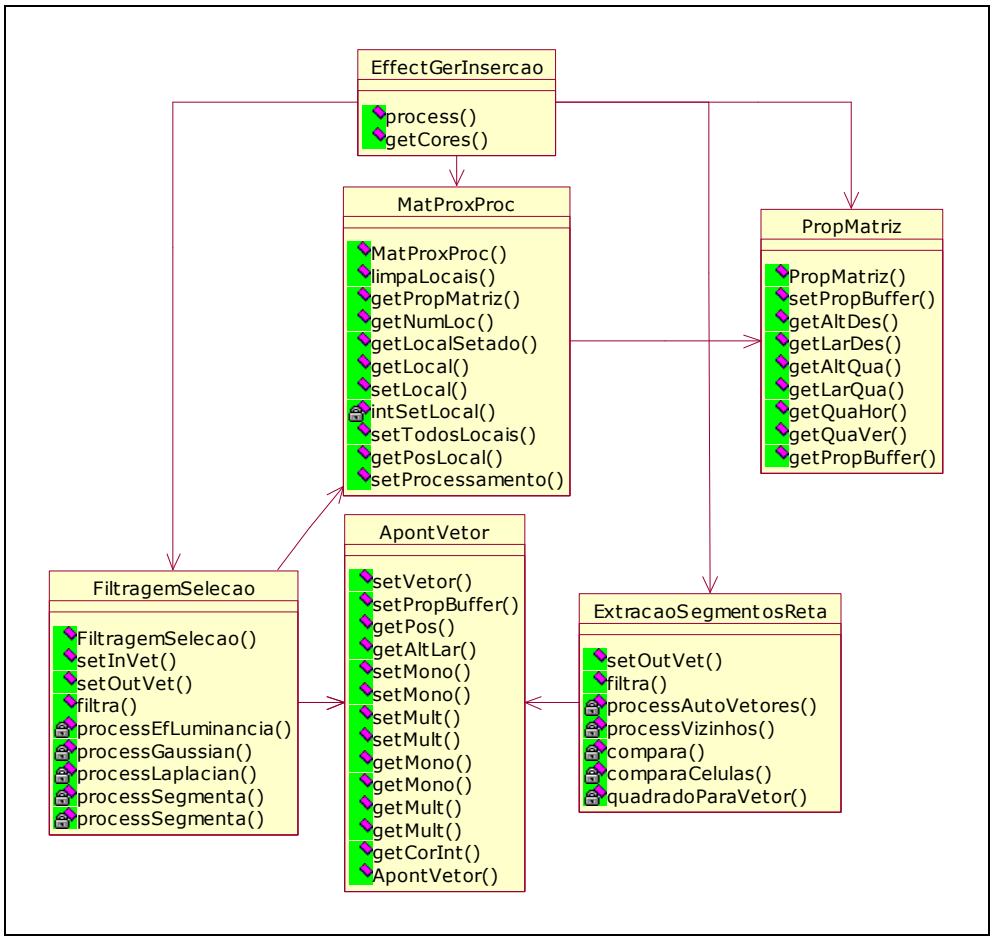
A seguir estão listados os diversos diagramas de classes que fazem parte da aplicação.

Figura 20 Diagrama de Classes Aplicação



A Figura 20 demonstra um diagrama de classes ao nível da aplicação. A classe truelogos representa o programa principal, contém a interface com o usuário. Os diversos métodos presentes nela são para manter a compatibilidade com as interfaces do JMF, que é a API utilizada como base para o processamento de vídeos no trabalho. A classe GerInsercao abre o vídeo de entrada e estabelece quais efeitos serão aplicados no vídeo na sua saída. A classe EffectBase é uma classe se preocupa apenas em implementar a interface Effect do JMF, deixando para suas classes filhas apenas a responsabilidade de realizar as transformações. Sua classe filha, EffectGerInsercao, realiza a Publicidade Virtual no frame atual do vídeo.

Figura 21 Diagrama de Classes EffectGerInsercao



A Figura 21 demonstra o diagrama de classes do EffectGetInsercao. A classe EffectGerInsercao é a responsável por processar cada frame do vídeo separadamente, através do método process(), especifico do JMF, ele é chamado para cada frame do vídeo.

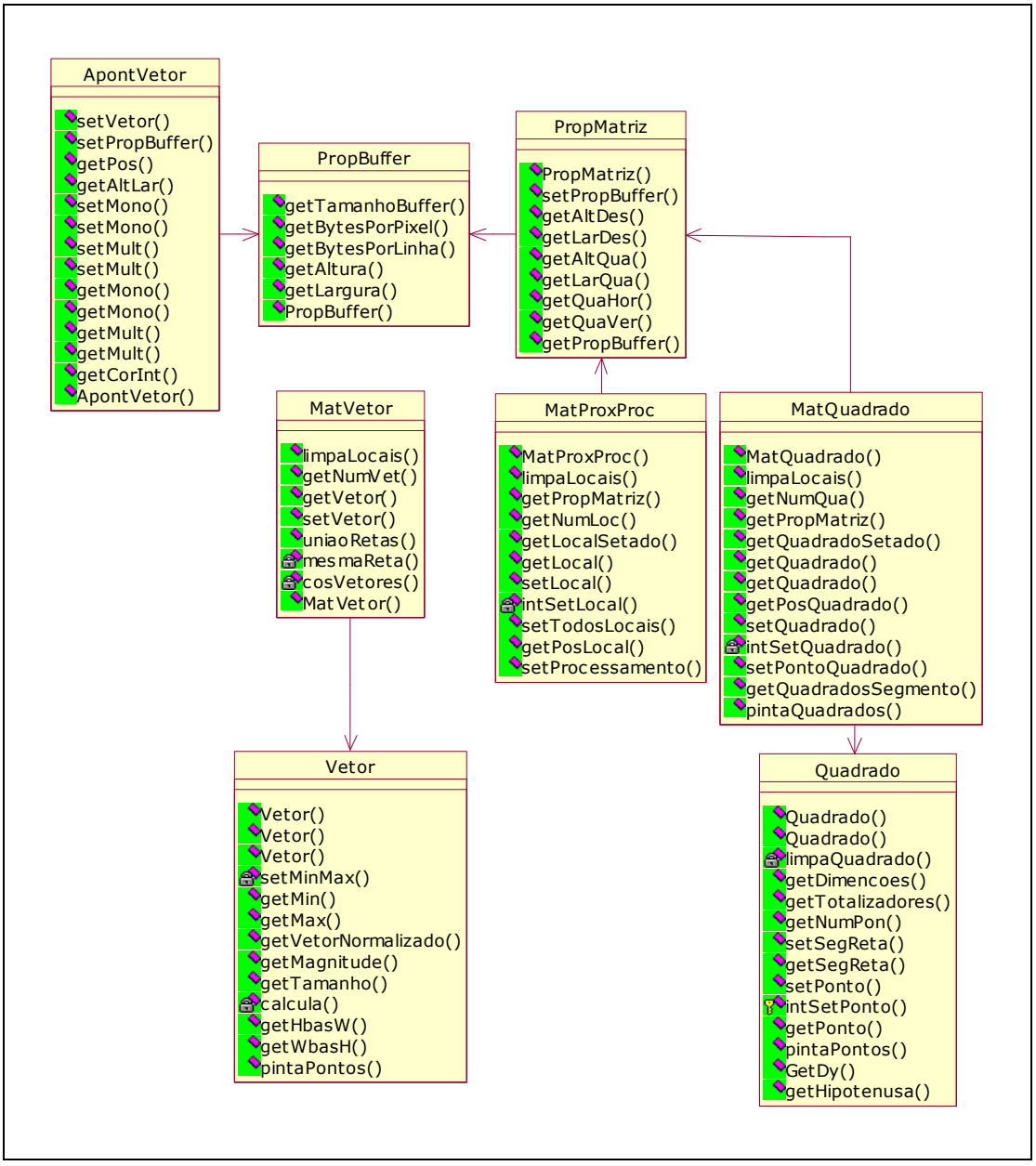
As classes FiltragemSelecao e ExtracaoSegmentosReta, fazem as transformações descritas nas secções 2.3.1 e 2.3.2 respectivamente. Elas utilizam a classe ApontVetor para lerem e gravarem dados do *frame* atual do vídeo.

A classe PropMatriz contém os parâmetros para quadricular a figura conforme a Figura 12. Além disso, foi criada uma área de descarte nas bordas do vídeo, pois as amostras de vídeo utilizadas apresentavam defeitos em seus cantos.

Já a classe MatProxProc informa à classe FiltragemSelecao os lugares da imagem que são aptos a conter as linhas do campo. As posições das linhas no *frame* atual devem

estar em uma posição ligeiramente diferente das posições do frame anterior, tornando assim desnecessário filtrar todo o *frame* a procura das linhas. O programa só filtra as áreas próximas às linhas encontradas no último *frame*.

Figura 22 Propriedades do Buffer e Vetores de Informação



A classe PropBuffer contém parâmetros do *buffer* como sua altura, largura e o tamanho do vetor das suas informações. As classes PropMatriz e ApontVetor, descritas na Figura 21 utilizam a classe PropBuffer para as suas respectivas finalidades.

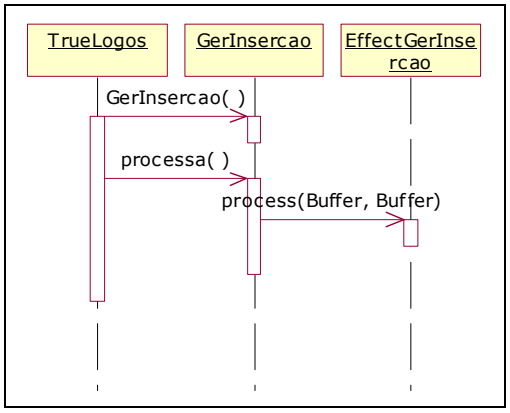
As classes MatVetor, MatProxProc e MatQuadrado são classes específicas para o armazenamento de conjuntos de informações. A principal diferença entre elas, tirando o tipo dos objetos que elas guardam, é que a MatProxProc e a MatQuadrado são modeladas conforme a classe PropMatriz, as posições que os seus elementos podem ocupar já estão previamente definidas e não é possível que dois elementos ocupem a mesma posição, enquanto a classe MatVetor não faz nenhuma restrição quanto a inclusão dos Vetores.

A classe Quadrado é responsável em armazenar os pontos dos quadrados durante as etapas descritas na seção 2.3.2.1 e 2.3.2.2. Já a classe Vetor armazenas os pontos da seção 2.3.2.4, que é a última etapa implementada neste protótipo.

3.2.3 DIAGRAMAS DE SEQÜÊNCIA

A seguir estão descritos os diversos diagramas de seqüência do protótipo, eles também foram divididos em diversos diagramas menores, para facilitar a interpretação dos mesmos.

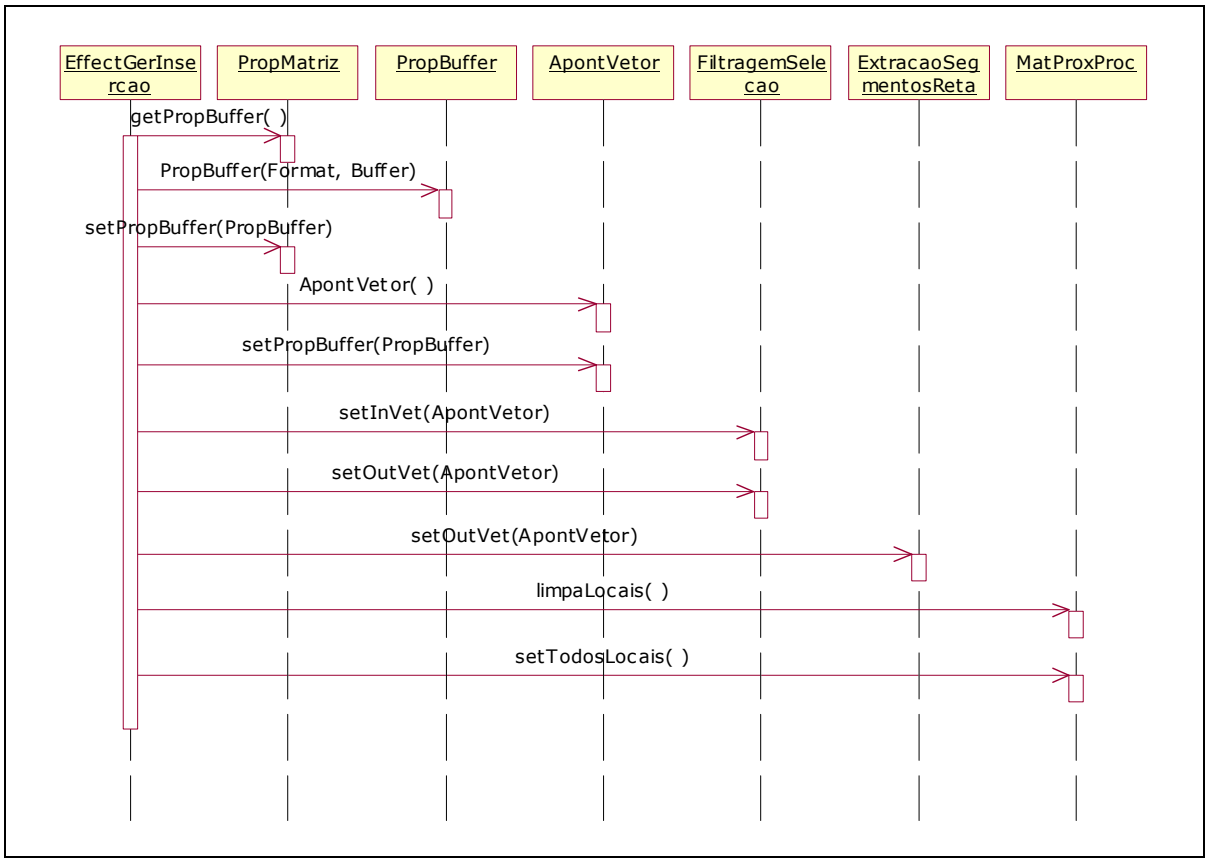
Figura 23 Diagrama de Seqüência da Aplicação



A Figura 23 demonstra o diagrama de seqüência da aplicação, a classe truelogos inicializa a classe GerInsercao, após isso ela pega o vídeo com a Publicidade Virtual realizada e o mostra no vídeo.

Já a classe GerInsercao, determina que será aplicado o efeito da classe EffectGerInsercao á cada frame do vídeo.

Figura 24 Diagrama de Seqüência da primeira passagem do método process do EffectGerInsercao



A Figura 24 demonstra os passos que são executados exclusivamente no primeiro *frame* do vídeo, para inicializar as diversas estruturas que dependem do formato do vídeo, que só é conhecido nesta etapa do processo.

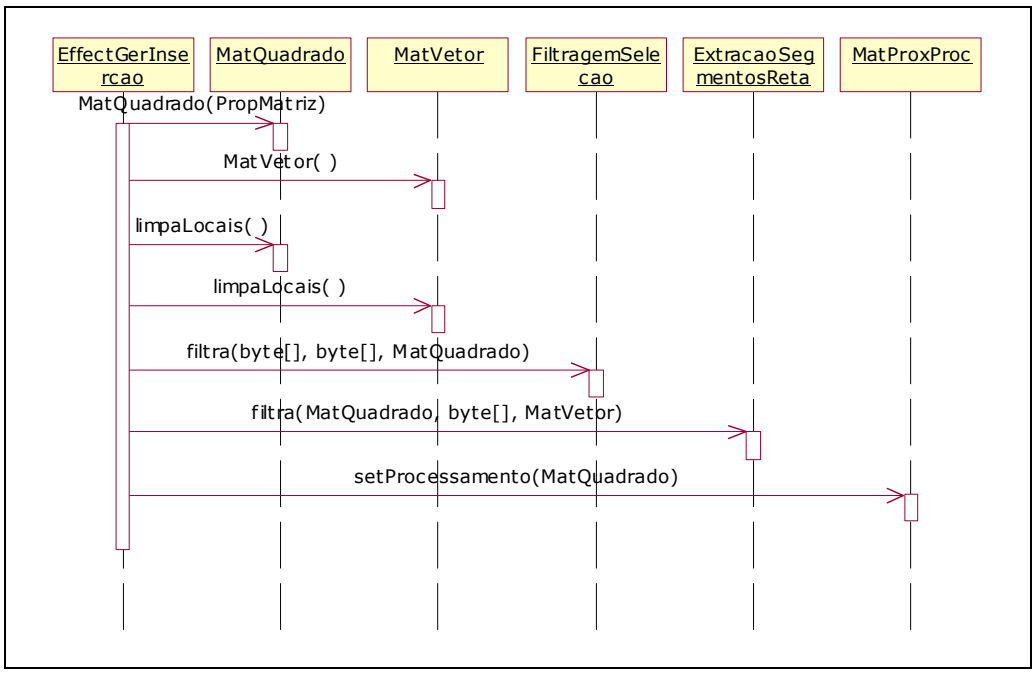
Para saber se o frame atual é o primeiro frame do vídeo, é verificado se a PropMatriz já possui a classe PropBuffer definida, se ainda não a possui, ela é criada e atribuída a PropMatriz.

São criadas duas instancias da classe ApontVetor, uma para leitura e outra para a gravação dos dados, isso é necessário para o cálculo da etapa descrita na seção 2.3.1.2 Realce de Linhas.

As duas instâncias da classe ApontVetor são passadas para as classes FiltragemSelecao e ExtracaoSegmentosReta.

A seguir é inicializada a classe MatProxProc, e parametrizado para ela processar todos os locais do frame, já que as posições aproximadas das linhas não são conhecidas.

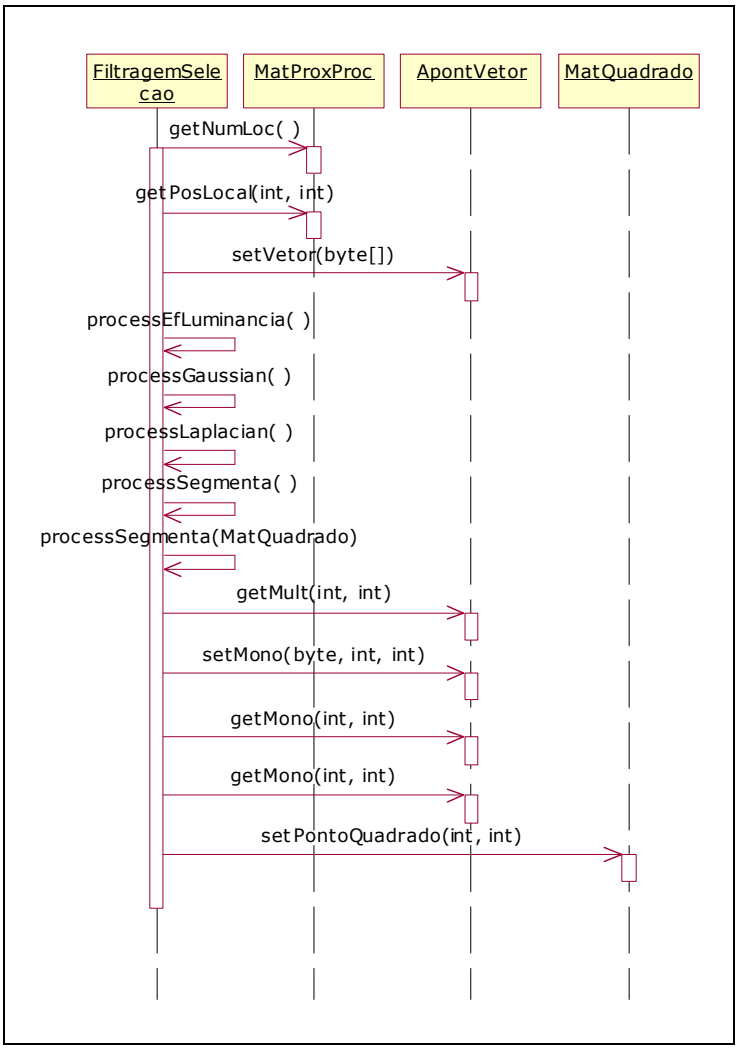
Figura 25 Diagrama de Seqüência do método process executado em cada frame do vídeo



A Figura 25 demonstra o processo que é executado para todos os frames do vídeo, independente de ser o primeiro ou não.

São criadas as classes MatQuadrado e MatVetor. Após isso, é feito o processamento propriamente dito, chamando o método filtra das classes FiltragemSelecao e ExtracaoSegmentosReta. Depois do *frame* processado, são indicados os locais do próximo *frame* que devem ser processados.

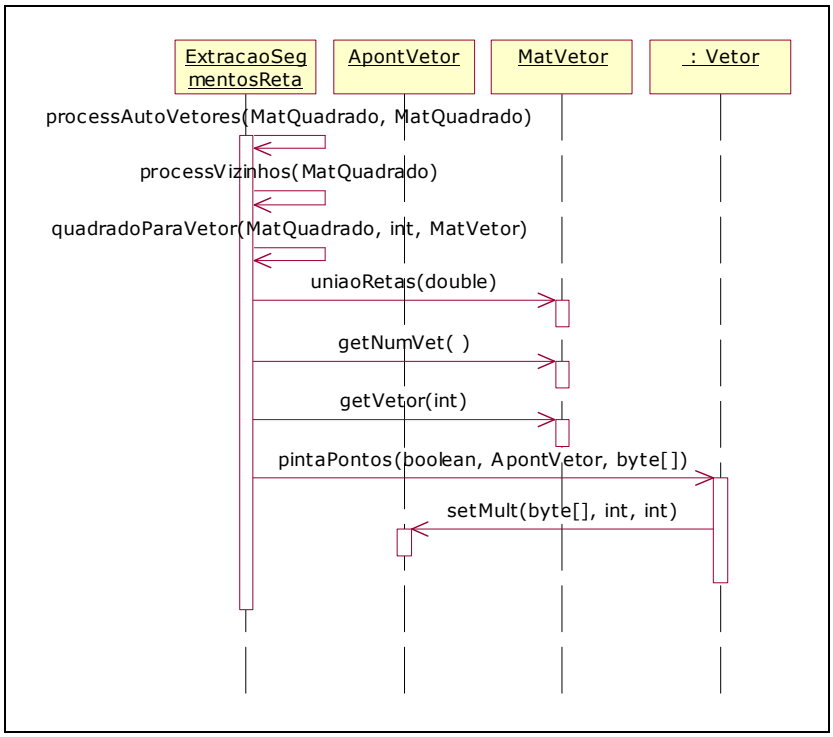
Figura 26 Diagrama de Seqüência do método filtra do objeto FiltragemSelecao



A Figura 26 demonstra o diagrama de seqüência do método filtra da classe FiltragemSelecao. Primeiro, verifica-se a quantidade de locais indicados para serem processados. Cada local corresponde a um quadrado da Figura 12. Para cada local indicado, são aplicados os filtros com as transformações de Luminância, Gaussian, Laplacian e Segmenta.

Os filtros, por sua vez, chamam os métodos getMult, setMono e getMono do ApontVetor. O filtro Segmenta seta os pontos selecionados na classe MatQuadrado para fazer a ligação com o ExtracaoSegmentosReta.

Figura 27 Método filtra do ExtracaoSegmentosReta



A Figura 27 demonstra o diagrama de seqüência do método filtra da classe ExtracaoSegmentosReta.

Primeiro é aplicado o método dos Autovalores, descrito na secção 2.3.2.1 para eliminar os quadrados que não formam nenhuma reta. Depois de filtrados os quadrados que formam retas, aplica-se o método processVizinhos para verificar as retas que ocupam vários quadrados, além de já enumerar as diversas retas presentes nos quadrados.

O próximo passo é transformar os quadrados em vetores. Isso é feito através do método quadradoParaVetor. Depois dos vetores criados, deve-se unir os vetores que tenham ângulos muito pequenos entre si, após isso, pinta-se os vetores encontrados.

3.3 IMPLEMENTAÇÃO

Este item aborda detalhes de como foi feita a programação do protótipo proposto por este trabalho.

3.3.1 FERRAMENTAS UTILIZADAS

Para o desenvolvimento deste trabalho foram utilizadas a linguagem Java e sua API Java Media FrameWork ou JMF. O JMF foi utilizado por que é distribuído gratuitamente em Sun (2003), é uma API especifica para o tratamento de vídeos e músicas, que engloba tratamento para utilizar os diversos formatos de vídeo existentes no mercado e também devido ao prestigio da linguagem Java na comunidade cientifica.

Como mencionado no último parágrafo, o JMF é uma API específica para a manipulação de vídeos e sons. Esta API foi desenvolvida para funcionar sobre a maquina virtual do Java Second Edition ou J2SE, logo, para o JMF poder funcionar deve-se ter instalado previamente o J2S e na máquina.

Informações sobre Java Second Edition são encontradas em Eckel (2000) e Sun (2003).

3.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

O protótipo possui a forma de um Appet, que é um tipo de programa específico para ser chamado a partir de páginas HTML. Dessa maneira toda a passagem de parâmetros e interface com o usuário foi abstraída do protótipo. No Quadro 19 encontra-se um exemplo de um código HTML que executa o protótipo.

Quadro 19 Exemplo de um código HTML que chama o protótipo

```
<HTML>
 <HEAD></HEAD>
 <BODY BGCOLOR="000000">
  <CENTER>
   <APPLET code = "TrueLogos.class" width ="500" height = "300">
    <param name=ORIGEM value="vídeo.mpg">
    <param name=TIPPRO value= "1">
    <param name=MINLOC value= "20">
    <param name=SOMLOC value= "1">
    <param name=TIPSAI value= "6">
    <param name=LIMIAR value= "15">
    <param name=AUTVAL value= "30">
    <param name=MINPON value= "8">
    <param name=AUTVIZ value= "15560">
    <param name=ANGMIN value= "0.995">
    <param name=NUMANG value= "5">
   </APPLET>
  </CENTER>
 </BODY>
</HTML>
```

Note que são passados vários parâmetros para o protótipo, no Quadro 20 é explicado resumidamente a função de cada um deles. Após isso eles serão explicados mais detalhadamente.

Quadro 20 Parâmetros da Aplicação

```
ORIGEM = Local em que o vídeo se encontra
TIPPRO = 0 = Processamento por proximidade geométrica
         1 = Processamento Total
MINLOC = Numero mínimo de locais aceitos para próximo processamento
SOMLOC = Quadrados laterais adicionados aos locais das linhas
TIPSAI = Visualização da Saída
         0 = Luminância
         1 = Gaussian
         2 = Laplacian
         3 = Filtragem
         4 = Autovalor
         5 = União com vizinhos
         6 = União de retas
LIMIAR = Valor do número de corte na Filtragem
AUTVAL = Valor mínimo aceito do Autovalor
MINPON = Numero mínimo de pontos em um quadrado no Autovalor
AUTVIZ = Valor mínimo aceito do Autovalor comparando com os vizinhos
ANGMIN = Valor mínimo do co-seno pra união de retas.
NUMANG = Quantidade de ângulos de segurança.
```

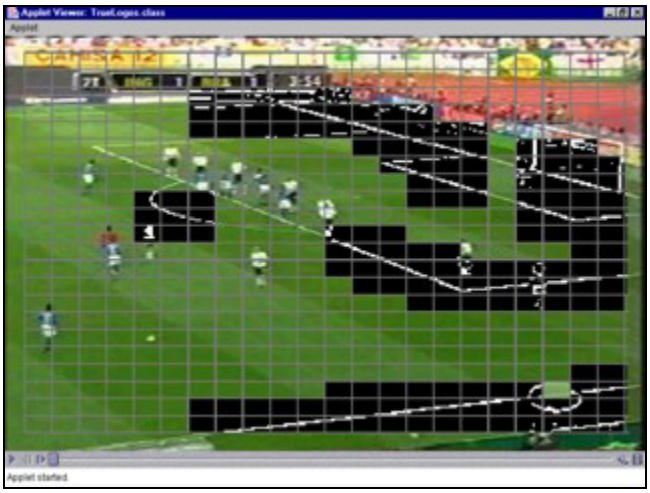
3.3.2.1 TIPOS DE PROCESSAMENTO

Existem dois tipos de processamento utilizados para especificar quais locais do *frame* atual serão processados em busca de linhas. O parâmetro TIPPRO define qual tipo de processamento será utilizado.

Se o valor do TIPPRO for igual a 0, será utilizado o Processamento por proximidade geométrica, que procura por novas linhas apenas nos locais próximos as linhas encontradas no frame anterior, com exceção ao primeiro frame, que é processado totalmente. As Figura 28 e Figura 29 demonstram exemplos do Processamento por proximidade geométrica.

Figura 28 Tipo de Processamento por proximidade geométrica

Figura 29 Tipo de Processamento por proximidade geométrica II

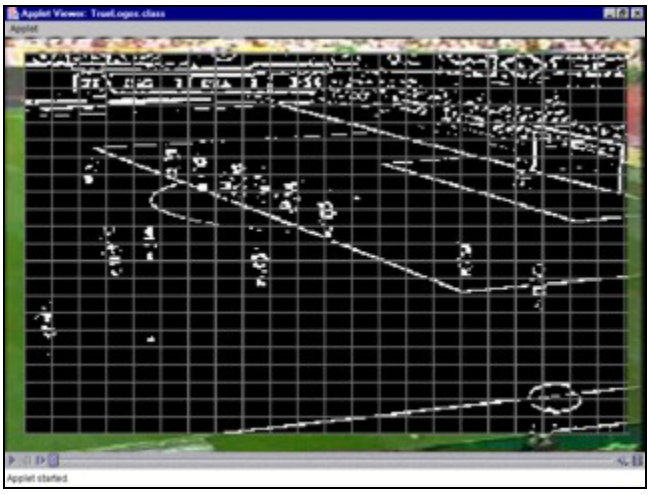


Na Figura 28, apenas os quadrados que estavam diretamente sobre as linhas foram marcados, já na Figura 29 os quadrados adjacentes aos quadrados com as linhas também foram marcados para processamento, isso ocorre por que na Figura 29 o parâmetro SOMLOC possui o valor 1, se possuísse valor 2, todos os quadrados com uma distância de até duas casas das linhas seriam marcados.

Outro parâmetro que influencia no Processamento por proximidade geométrica é o MINLOC, este parâmetro foi criado porque em algumas situações nenhum quadrado é marcado, não havendo mais processamento deste ponto em diante do vídeo. Assim, sempre que o número de quadrados marcados for inferior ao valor de MINLOC, todos os quadrados serão novamente marcados.

Se o valor do TIPPRO for diferente de 0, será utilizado o Processamento Total, que varre toda a imagem em cada frame do quadro. Ele é mais lento que o Processamento por proximidade geométrica, porém todas as linhas são pegas neste processamento. A Figura 30 demonstra um exemplo de um frame com Processamento Total.

Figura 30 Tipo de Processamento Total



3.3.2.2 VISUALIZAÇÃO DA SAÍDA

O parâmetro TIPSAI controla as opções de visualizações disponíveis. Foi utilizado o tipo de processamento total nas imagens a seguir, pois esta visualização é a que apresenta os melhores resultados para fins didáticos.

A Figura 31 mostra a visualização da Luminância. O parâmetro TIPSAI vale zero para se obter esse resultado.

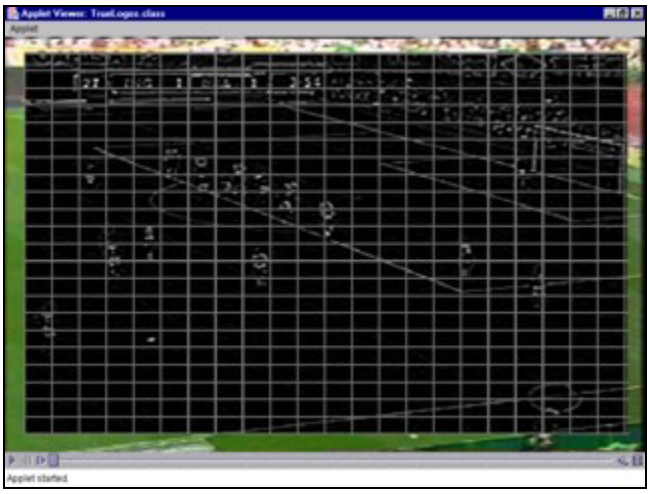
Figura 31 Visualização 0, Luminância

A Figura 32 mostra a visualização do filtro de Gaussian. O parâmetro TIPSAI vale um para se obter esse resultado.Note que a Figura 32 está mais fosca do que a Figura 31, devido ao zoom, esta diferença não é tão perceptível nas figuras.

Figura 32 Visualização 1, Gaussian

A Figura 33 mostra a visualização do Filtro de Laplacian. Note que neste estágio estão visíveis os locais aonde existem diferenças de tonalidade com relação aos pixels vizinhos, indicando assim a presença de um objeto neste local.

Figura 33 Visualização 2, Laplacian



A Figura 34 mostra o resultado do filtro de Filtragem, os Pixels cujo valor escalar forem maiores que o valor do parâmetro LIMIAR, são pintados de branco, os demais são pintados de preto.

Figura 34 Visualização 3, Filtragem

A Figura 35 mostra o resultado dos Autovalores, os quadrados que estão marcados são os que possuem uma quantidade de pontos superior ao parâmetro MINPON e o seu autovalor é superior ao parâmetro AUTVAL.

Figura 35 Visualização 4, Autovalores

A Figura 36 demonstra o resultado da união dos quadrados com seus vizinhos. Os quadrados que possuem a mesma cor das linhas foram unidos nesta etapa do processo. Para isso, o valor do Autovalor da união dos seus pontos deve ser superior ao parâmetro AUTVIZ.

Figura 36 Visualização 5, União com vizinhos

A Figura 37 mostra a união das retas que tem a mesma direção. O parâmetro ANGMIN contém o co-seno da diferença máxima permitida entre as duas retas. O parâmetro NUMANG contém o número de vezes que a união é feita com ângulos superiores , para amenizar o problema das Aberrações. O Quadro 21 demonstra um exemplo da utilização desses parâmetros.

Figura 37 Visualização 6, União das Retas

Quadro 21 Exemplificação dos parâmetros ANGMIN e NUMANG

```
ANGMIN = 0.995
NUMANG = 5

Verificação 1:
Valor Angulo na união = 0.999

Verificação 2:
Valor Angulo na união = 0.998

Verificação 3:
Valor Angulo na união = 0.997

Verificação 4:
Valor Angulo na união = 0.996

Verificação 5:
Valor Angulo na união = 0.995
```

3.4 RESULTADOS E DISCUSSÕES

Um dos maiores problemas encontrados na durante execução deste trabalho foi o de encontrar valores válidos para os parâmetros de entrada do protótipo. Partiu-se do princípio que é possível encontrar valores genéricos, válidos para todos os vídeos de jogos de futebol utilizados, porém, estes valores não foram encontrados. Existem muitos parâmetros que influenciam na obtenção dos valores de entrada, como a qualidade das filmagens e o nível de aberrações encontrados nos mesmos, geradas pelas câmeras utilizadas. Este problema, atribuição dos parâmetros de entrada, tornou-se mais evidente na união dos segmentos de reta.

As aberrações distorcem de maneira significativa as retas encontradas na operação de união dos segmentos de reta, descrita na seção 2.3.2.4. Devido às aberrações de imagem, retas paralelas são unidas formando novas retas, que eram inexistentes no vídeo original. Uma maneira de minimizar este problema é diminuir a diferença aceita entre os ângulos dos segmentos de reta, porém, se esta diferença diminuir demais, os segmentos de reta não serão mais unidos. Para resolver este problema foi criado um método que faz a união de retas várias vezes, unindo gradativamente retas que estejam próximas de serem paralelas, conforme o Quadro 21.

4 CONCLUSÕES

. A idéia inicial proposta para este trabalho de conclusão de curso era implementar um protótipo que realizasse o processo completo de Publicidade Virtual, desde a inferência do sistema de coordenadas até a inserção dinâmica, quadro a quadro de imagens estáticas no vídeo. No decorrer da elaboração do trabalho, o problema mostrou-se demasiadamente complexo para conclusão no cronograma previsto inicialmente. Por isso, apenas as etapas iniciais do desenvolvimento de um sistema para publicidade virtual foram abordadas. Até o presente momento, foram implementadas, na forma de um protótipo, as etapas *Filtragem e Seleção* e a *Extração dos Segmentos de Reta*, porém a pesquisa para a implementação das demais etapas continua em desenvolvimento.

Em relação aos futuros problemas que se apresentam frente à continuidade deste trabalho de pesquisa, encontra-se a incerteza quanto à eficiência e correção do método de calibragem de câmeras em situações reais, visto que o objetivo do sistema é funcionar em transmissões ao vivo de jogos de futebol em TV aberta.

Outro problema a ser avaliado futuramente é a questão do desempenho computacional, pois ainda não foi possível comprovar experientalmente se a linguagem Java demonstra desempenho suficiente para realizar a calibragem de câmeras em tempo real pelo método utilizado neste trabalho. O protótipo apresentado no capítulo 3 apresentou uma taxa de reconhecimento de linhas a 3 quadros por segundo para imagens de vídeo com resolução de 400x200 pixels em um computador com processador 1.8 Gigabytes redando o sistema operacional Windows XP com 256Mb de memória RAM. Para realizar a publicidade virtual com sinal de vídeo nos padrões usados pelas emissoras de TV, NTSC ou Pal-M, o protótipo deve ser capaz de calibrar a câmera e inserir as imagens dinamicamente em uma taxa maior do que 15 quadros por segundo, em imagens de aproximadamente 500 por 500 *pixels*.

4.1 EXTENSÕES

Como sugestões para trabalhos futuros, além da implementação completa da Publicidade Virtual em jogos de Futebol, que era o objetivo inicial da proposta de trabalho,

sugere-se estender a implementação para diversos outros esportes tais como o Tênis e o Vôlei. Esta tarefa requer uma adaptação do método apresentado, no sentido de contemplar as particularidades geométricas dos campos ou quadras diferentes do campo de futebol. Em resumo, há a necessidade de implementar diferentes árvores e regras de interpretação, conforme mostrado na seção 2.3.3.

O presente trabalho apresenta subsídios para desenvolvimento de outros tipos de sistemas que utilizam Computação Gráfica na visualização e interpretação de cenas de esportes, pois tendo-se calibrado a câmera, é possível aplicar diversos outros efeitos a imagem, como calcular a distância entre dois pontos do campo, criando um sistema do tipo "tira-teima" ou ainda modelar toda a cena para um ambiente virtual.

Um assunto que ainda não foi explorado na bibliografia revisada consiste no pré-processamento da imagem para estimar convenientemente os parâmetros de entrada, de forma a minimizar os problemas apresentados na seção 3.4. Esta necessidade surgiu na fase final de realização deste trabalho de conclusão de curso, e ainda representa um assunto completamente inexplorado pelo autor. Não há ainda indícios sobre soluções viáveis para o problema, entretanto seriam de grande utilidade na implementação de um futuro sistema comercial.

5 REFERÊNCIAS BIBLIOGRÁFICAS

BERTULANI, Carlos A. **Luz e Cor**. [S.1.] [2000]. Disponível em: <http://www.if.ufrj.br/teaching/luz/cor.html/>. Acesso em: 26 mai 2003.

ECKEL, Bruce. **Thinking in Java.** 2. ed. New Jersey: Prentice-Hall, 2000. 1129 p.

FOLEY, James D. **Computer graphics**: principles and practice. 2. ed. New York: Addison-Wesley, 1990. 1175 p.

FURLAN, Davi. **Modelagem de objetos através da UML** – the unified modeling language. São Paulo: Makron Books, 1998. 329 p.

GBDI, Grupo de Base de Dados e Imagens. **Apostila de computação Gráfica**. [S.l.], [2003?]. Disponível em: <http://gbdi.icmc.sc.usp.br/membros/index.html/>. Acesso em: 26 mai. 2003.

GOMES, Paulo César Rodacki. **Desmistificando a publicidade virtual**. Guia informativo sobre a tecnologia de publicidade virtual. Publicação interna da Rede Globo de Televisão/Divisão de Engenharia de Multimídia (DIEM)/Central Globo de Engenharia (CGE). Rio de Janeiro, 1999.

HIPR2, Hypermedia Image Processing Reference. **Image Processing Learning Resources** [S.l.], [2000]. Disponível em: <http://www.dai.ed.ac.uk/HIPR2/hipr_top.htm/> Acesso em: 26 mai. 2003.

MORTENSON, Michael E. **Mathematics for computer graphics applications**. New York: Industrial Press, 1999.

SUN MICROSYSTEMS. **The Source for Java Technology**, [S.l.], [2003?]. Disponível em: <http://java.sun.com/>. Acesso em: 3 abr. 2003.

SZEMBERG, Flávio. **Acompanhamento de cenas com calibração automática de câmeras.** 2001. 144 f. Tese (Doutorado em Ciências em Informática) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

TONIETTO, Leandro. **Análise de algoritmos para Chroma-Key**. [S.1.] [2000]. Disponível em: <http://www.inf.unisinos.br/~cglab/equipe/leandrot/tc.pdf/>. Acesso em: 25 mai. 2003.

APÊNDICE A – MATRIZES HOMOGÊNEAS DE TRANSFORMAÇÃO

Para realizar todas as transformações geométricas dos elementos presentes neste trabalho, foram utilizadas as Matrizes Homogêneas de Transformação, descritas por Mortenson (1999), por Foley (1990) e por Gbdi(2003).

Seu princípio básico consiste em que, cada transformação geométrica possui a sua respectiva matriz. Em Quadro 22, Quadro 23, Quadro 24, Quadro 25 e Quadro 26 podem ser encontradas diversas matrizes, uma para cada transformação.

Quadro 22 Rotação no eixo z

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -sin\theta & 0 & 0 \\ sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x^1 & y^1 & z^1 & 1 \end{bmatrix}$$

Quadro 23 Rotação no eixo y

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x^1 & y^1 & z^1 & 1 \end{bmatrix}$$

Quadro 24 Rotação no eixo x

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\gamma & -sin\gamma & 0 \\ 0 & sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x & y & z & 1 \end{bmatrix} = \begin{bmatrix} x^1 & y^1 & z^1 & 1 \end{bmatrix}$$

Quadro 25 Escala

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x^1 & y^1 & z^1 & 1 \end{bmatrix}$$

Quadro 26 Translação

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ p & q & r & 1 \end{bmatrix} = \begin{bmatrix} x^1 & y^1 & z^1 & 1 \end{bmatrix}$$

Como todas possuem as matrizes possuem as mesmas dimensões, é possível combinar diversas transformações em uma única matriz, criando assim uma matriz que faz diversas transformações geométricas de uma só vez, evitando assim desperdícios de processamento.