UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE SOFTWARE PARA INSERÇÃO DE PUBLICIDADE VIRTUAL EM VÍDEOS DE JOGOS DE FUTEBOL

JEAN GEARD HAGEN

BLUMENAU
2005

2005/1-26

JEAN GEARD HAGEN

PROTÓTIPO DE SOFTWARE PARA INSERÇÃO DE PUBLICIDADE VIRTUAL EM VÍDEOS DE JOGOS DE FUTEBOL

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Paulo César Rodacki Gomes - Orientador

BLUMENAU
2005

2005/1-26

PROTÓTIPO DE SOFTWARE PARA A INSERÇÃO DE PUBLICIDADE VIRTUAL EM VÍDEOS DE JOGOS DE FUTEBOL

Por

JEAN GEARD HAGEN

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Paulo César Rodacki Gomes, Dr. – Orientador, FURB
Membro:	Prof. Mauro Mattos, Dr. – FURB
Membro:	Prof. Francisco Adell Péricas, Ms. - FURB

Blumenau, 03 de junho de 2005

Dedico este trabalho a todos os amigos, especialmente àqueles que me ajudaram diretamente na realização deste, em especial a meus pais que entenderam a minha ausência e à minha irmã, que procurou me auxiliar em todos os momentos.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, que mesmo longe, sempre esteve presente.

Aos meus amigos, pelos empurrões e cobranças.

Ao meu novo amigo Everton Elvio Koser, por sua grande ajuda em vários momentos de dificuldade no desenvolvimento do trabalho.

Ao meu orientador, Paulo César Rodacki Gomes, por ter acreditado na conclusão deste trabalho.

Os bons livros fazem “sacar” para fora o que a pessoa tem de melhor dentro dela.

Lina Sotis Francesco Moratti

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo de software para calibração automática de câmeras em vídeos de jogos de futebol. O método utilizado para obtenção dos parâmetros para calibração de câmeras consiste em descobrir as linhas que representam as delimitações do campo de futebol bem como da grande e da pequena área, de forma automática. É apresentada uma técnica para classificação das linhas para correlacionamento com um modelo virtual de campo de futebol. Com isto, espera-se ser possível reproduzir a projeção em perspectiva feita pela câmera ao capturar as imagens do jogo, de forma a permitir a inserção de novas imagens no vídeo com a mesma transformação de projeção da câmera ou até mesmo reconstituir partes do vídeo que não aparecem, mais que desta forma são passíveis de serem deduzidas. O protótipo foi implementado na linguagem Java.

Palavras-chave: Computação gráfica. Reconhecimento e interpretação de imagens. Calibração de câmeras.

ABSTRACT

This work presents the development of an archetype of software for automatic calibration of cameras in videos of soccer games. The method used for attainment of the parameters for calibration of cameras consists of discovering the lines that represent the delimitations of the field of soccer as well as of the great e of the small area, of automatic form. One technique for classification of the lines for correlacionamento with a virtual model of soccer field is presented. With this, one expects to be possible to reproduce the done perspective projection for the camera when capturing the images of the game, of form to allow the insertion of new images in the video with the same transformation of projection of the camera or even though to reconstitute parts of the video that do not appear, more than of this form they are passíveis to be deduced. The archetype was implemented in the Java language.

Key-Words: Graphical computation. Recognition and interpretation of images. Calibration of cameras.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente, os profissionais de televisão vêem pensando em alternativas para a veiculação de anúncios publicitários pois é cada vez mais difícil prender a atenção do telespectador para a apresentação de comerciais da forma tradicional, que é a exibição de clipes comerciais nos intervalos da programação. Cada vez mais são necessárias novas técnicas de publicidade para anunciar determinado produto sem que o telespectador mude de canal.

A publicidade virtual consiste numa técnica nova em comunicação, pois o local onde se apresenta a publicidade é a mesma da cena onde ocorre a ação. A publicidade virtual apresenta-se de tal forma a parecer que faz parte da ação, com a vantagem de o telespectador estar concentrado na ação. Com isso, espera-se que a mensagem publicitária seja absorvida pelo telespectador com muito mais êxito.

Segundo Gomes (1999, p. 5), “Publicidade Virtual consiste basicamente na inserção eletrônica de anúncios publicitários dentro da imagem de vídeo. A idéia é que esta inserção seja feita de tal maneira que o tele-espectador tenha a impressão de que os anúncios fazem parte da cena realmente.” As figuras 1 e 2 ilustram um exemplo de aplicação deste conceito.

Fonte : Gomes (1999, p. 5)

Figura 1 - Imagem sem o uso da publicidade virtual

Fonte : Gomes (1999, p. 6)

Figura 2 - Imagem com o uso da publicidade virtual

A figura 1 representa um quadro do jogo de futebol sem o uso da publicidade virtual, já a figura 2 é o mesmo quadro com o uso da publicidade virtual.

Para cada quadro do jogo, se deve realizar um processo de calibração de câmera, isto é, descobrir os parâmetros da câmera que está gerando o quadro. Para isso, Szenberg (2001, p.7) demonstra que é necessário reconhecer alguma parte do campo através da imagem de entrada. Com isso feito são definidos os parâmetros de projeção de imagem da câmera real. Este processo é realizado para cada quadro devido à câmera poder se movimentar na imagem, com esse movimento da câmera as posições para a inserção da publicidade virtual mudam. Feito isso, é possível inferir um sistema de coordenadas para o campo de futebol e, assim, determinar as coordenadas para a inserção virtual de forma que ela pareça fazer parte da cena capturada em vídeo.

Levando-se em consideração estes fatos, busca-se nesta proposta continuar o trabalho relatado em Koser (2003), o qual trata do reconhecimento automático de linhas de campo de futebol em arquivos de vídeo para publicidade virtual, o qual chegou no estágio de reconhecimento de segmentos de reta, que são candidatos a serem retas do campo de futebol. Este trabalho agregará a capacidade de organizar estes segmentos de reta em segmentos de

reta longos, e posteriormente o reconhecimento destes segmentos de reta em relação às linhas do campo, isto é, descobrir qual segmento representa a linha lateral do campo e assim sucessivamente, para as outras linhas do campo. Pretende-se desenvolver um protótipo de *software* que seja capaz de organizar e reconhecer os segmentos de reta. Para isso, deve-se estimar a posição e as dimensões dessas imagens em cada quadro do vídeo, tendo como elementos para cálculo desses parâmetros apenas as imagens em vídeo do jogo de futebol.

1.1 OBJETIVOS DO TRABALHO

O objetivo do trabalho é implementar um protótipo de *software* capaz de fundir em um vídeo com lances de partida de futebol, imagens digitais que representariam o papel de publicidade virtual.

Como objetivos específicos tem-se:

a) conseguir identificar na imagem do campo de futebol as linhas da lateral, do final do campo, e as linhas que delimitam a pequena e a grande área;

b) inferir os parâmetros de definição da câmera em relação à imagem do campo de futebol;

c) distorcer a imagem (logo) que será inserida na imagem do campo de futebol para que a imagem (logo) a ser inserida pareça ser real;

d) inserir a imagem (logo) distorcida na posição correta dentro da imagem do campo de futebol.

1.2 ESTRUTURA DO TRABALHO

O capítulo 2 apresenta a fundamentação teórica na qual este trabalho é embasado. Inicialmente, é apresentada uma introdução sobre publicidade virtual, com uma comparação entre o modelo tradicional e o que se busca implementar. Em seguida são apresentados alguns aspectos relevantes do trabalho de Koser (2003), e depois os temas inerentes à realização deste trabalho como união de segmentos de reta, reconhecimento e interpretação, câmeras de vídeo, calibragem de câmeras e por último trabalhos correlatos.

No capítulo 3 são abordados os processos envolvidos no desenvolvimento do protótipo, tais como a definição dos principais requisitos do sistema, especificação com os diagramas de casos de uso e diagrama de atividades, técnicas e ferramentas utilizadas, a operacionalidade do protótipo e uma explanação sobre os resultados obtidos.

O capítulo 4 apresenta as conclusões sobre a aplicação das técnicas propostas, bem como uma discussão a respeito de propostas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta primeira parte da fundamentação teórica é feita uma relação entre a técnica tradicional de publicidade virtual, e a técnica usada neste trabalho. A segunda parte trata do trabalho feito por Koser (2003), já que o presente trabalho trata de uma continuação do trabalho de Koser. A terceira parte aborda outros assuntos inerentes à realização deste trabalho, tais como: extração de segmentos de reta, câmeras de vídeo, calibragem de câmeras, reconhecimento e interpretação, acompanhamento de cenas e uma descrição dos trabalhos correlatos. A última parte comenta de trabalhos correlatos.

2.1 PUBLICIDADE VIRTUAL

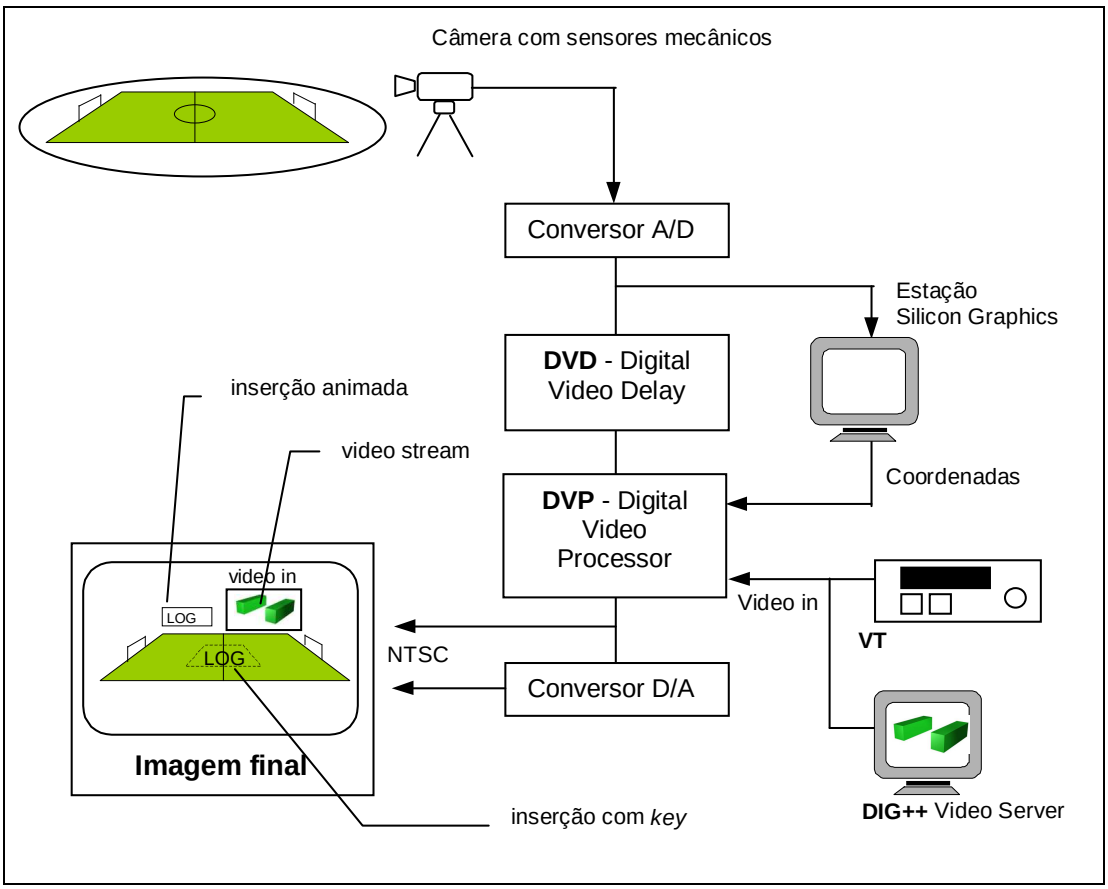
De acordo com Gomes (1999, p. 11), publicidade virtual consiste em um método de publicidade utilizando o aparato técnico de televisão existente no qual o anunciante escolhe partes da cena onde ocorrerá a publicidade.

2.1.1 EQUIPAMENTO NO MODELO TRADICIONAL

Para a realização da publicidade virtual do modo tradicional, é utilizada uma série de equipamentos como mostra a figura 3. De acordo com Gomes (1999) processo é iniciado com a captura das imagens por uma câmera de TV adaptada para conter uma série de sensores mecânicos adaptados à sua lente e à cabeça de seu tripé. Estes sensores geram dados contínuos do movimento de *pan* (movimento horizontal), *tilt* (movimento vertical), *zoom* e foco de câmera, se a câmera não for digital, as imagens captadas são digitalizada por um

conversor A/D (analógico digital). Do Conversor A/D partem dois sinais digitais de vídeo, um principal e um secundário. O sinal principal é o sinal que contém as cenas filmadas. O secundário por sua vez, é um sinal auxiliar usado pelos aplicativos de configuração e monitoramento do sistema.

De acordo com Gomes (1999) o sinal principal segue para um Digital Vídeo Delay (DVD) onde sofre um pequeno retardo equivalente a dois *frames*. Este tempo equivale ao tempo necessário para que a estação calcule as coordenadas para a inserção da publicidade, e forneça essas informações em sincronia com o sinal principal de vídeo. Além do sinal secundário de vídeo a estação recebe a leitura dos sensores da câmera, que serão usados no cálculo das coordenadas para a inserção.



Fonte : Gomes (1999, p. 12)

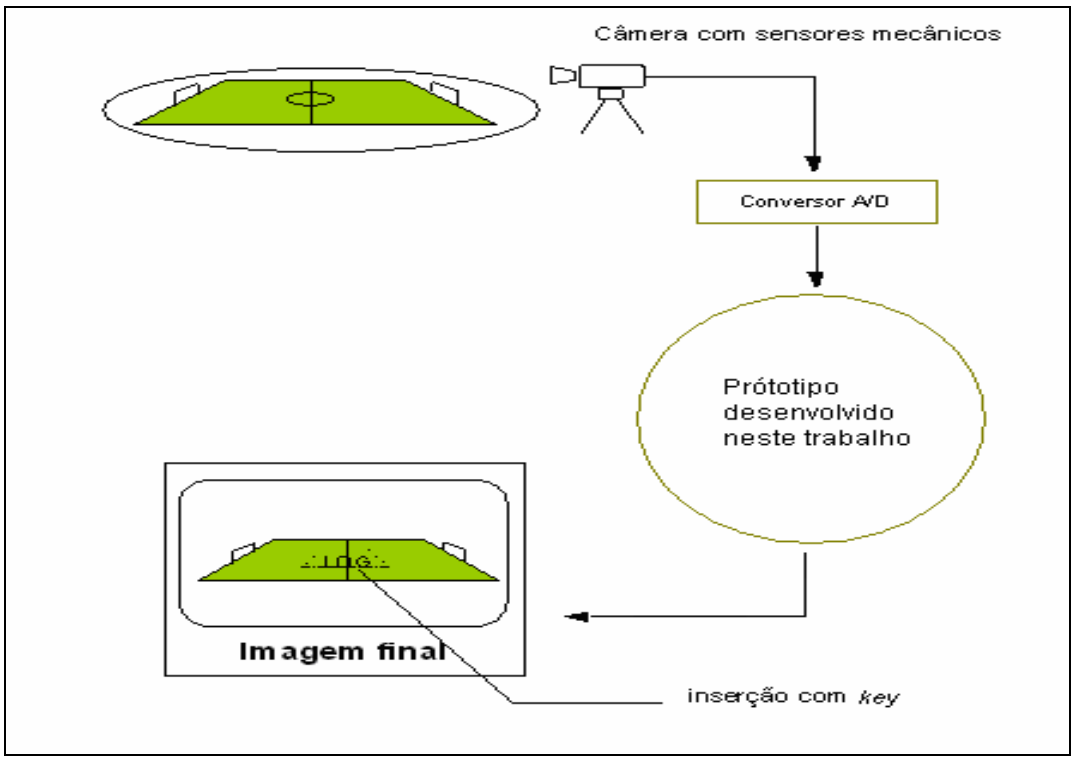
Figura 3 - Representação esquemática do equipamento

De acordo com Gomes (1999) o DVP (Digital Vídeo Processor) é um computador gráfico dedicado ao processamento de publicidade virtual. Este computador recebe da estação as coordenadas para cada *insert* de publicidade virtual. Em sua memória ficam armazenadas as imagens e animações a serem inseridas no sinal de vídeo. Com estas informações o DVP manipula o sinal de vídeo e insere quadro a quadro as imagens do anúncio dentro da imagem do tema capturado pela câmera.

Depois deste processo, o vídeo pode ser convertido para o sinal analógico ou ser transmitido no formado digital.

2.1.2 EQUIPAMENTO NO SISTEMA PROPOSTO

No modelo proposto no presente trabalho, a maioria dos equipamentos do modelo tradicional é substituído pelo protótipo de *software* como mostra a figura 4.

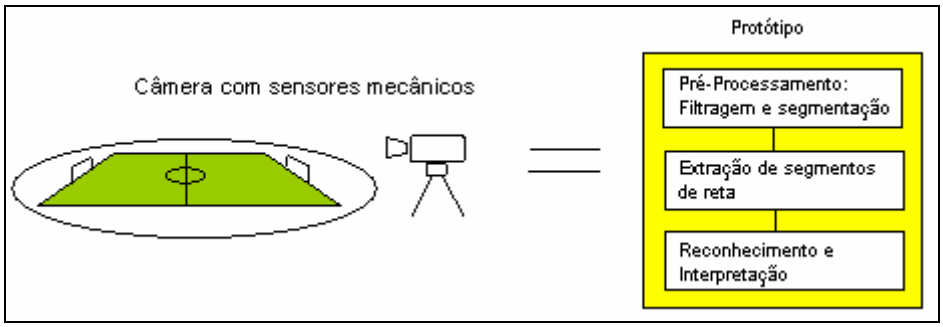


Fonte : Adaptado Gomes (1999, p. 12)

Figura 4 - Representação esquemática do equipamento do protótipo.

O primeiro passo no protótipo é conseguir substituir os sensores da câmera por um conjunto de algoritmos capazes de gerar o sinal secundário, como no modelo tradicional. Para o sistema proposto, esse sinal secundário seria substituído por um conjunto de pontos cuja localização na imagem é conhecida.

Esta etapa engloba três algoritmos, conforme ilustrado na figura 5: pré-processamento, filtragem e segmentação, extração de segmentos de retas e reconhecimento e interpretação.



Fonte : Adaptado Gomes (1999, p. 12)

Figura 5 – Correlação entre o sistema tradicional e o protótipo.

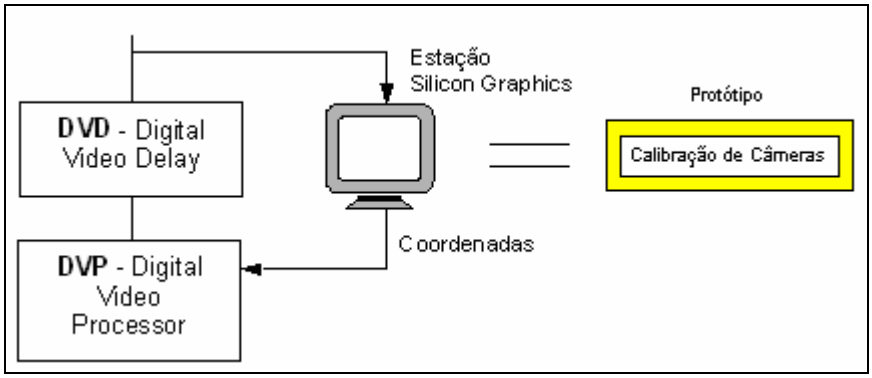
O pré-processamento é a fase onde são aplicados filtros na imagem de entrada, para realçar nesta imagem objetos já pré-conhecidos. A idéia é que estes objetos possam auxiliar o processo de reconhecimento e interpretação. No caso da imagem do campo de futebol, sabe-se que, no campo, existem linhas que delimitam a grande área e as laterais do campo. Essas linhas são um conjunto de objetos já pré-conhecidos para o sistema, sua localização na imagem pode fornecer informações equivalentes à que serão obtidas pelos sensores externos.

O resultado da fase de pré-processamento é um conjunto de pontos. Logo após, é executado um algoritmo de extração de segmentos de reta a partir do conjunto de pontos.

Na última etapa da primeira fase é executado o algoritmo de reconhecimento e interpretação, o qual recebe o conjunto de retas da etapa anterior, e faz a correlação entre as retas encontradas e as retas pré-conhecidas em modelo virtual de campo de futebol.

Com o reconhecimento dessas retas, procura-se determinar um conjunto de pontos cuja localização na imagem é conhecida.

A segunda etapa do processo a ser realizada pelo protótipo é a calibração de câmera, conforme figura 6.



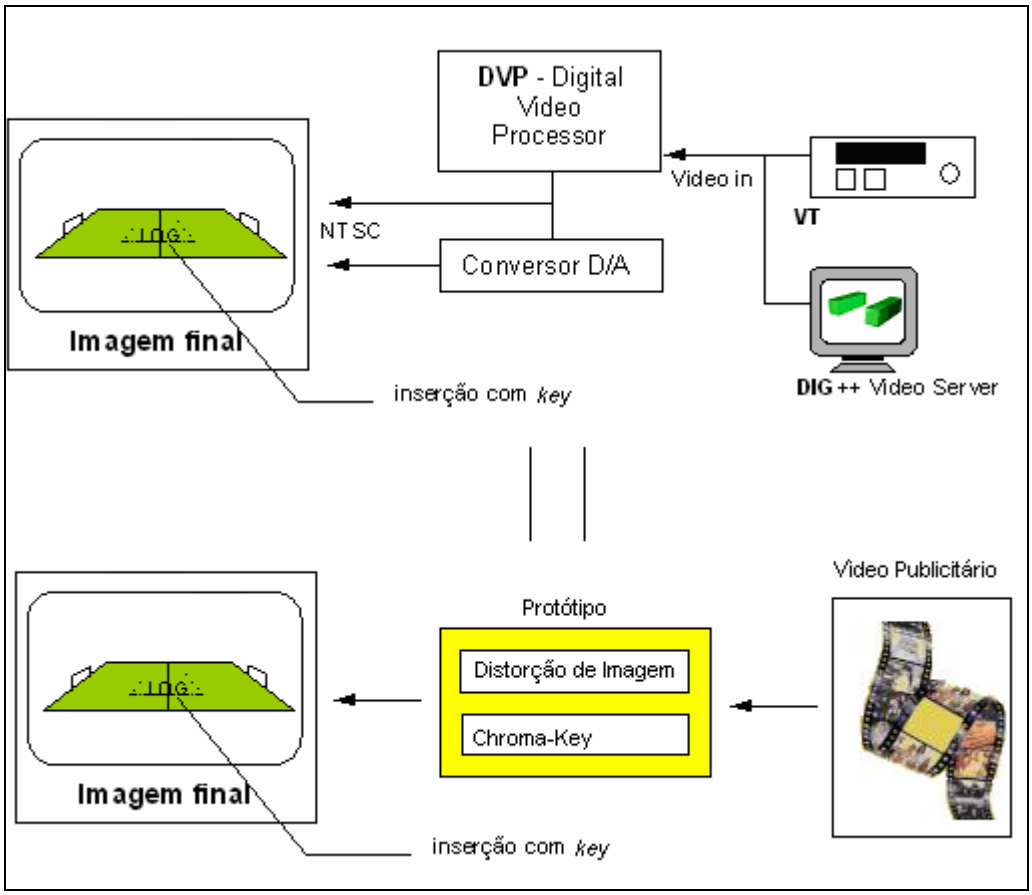
Fonte : Adaptado Gomes (1999, p. 12)

Figura 6 - Correlação entre o sistema tradicional e o protótipo

Com o conjunto de pontos da etapa anterior é possível inferir valores para determinar parâmetros de configuração da câmera virtual equivalentes aos da câmera real.

Após a determinação dos parâmetros de câmera, é possível aplicar a transformação de projeção de forma a inserir mais elementos na cena com a mesma distorção de perspectiva feita pela câmera real. Este processo tem como saída também uma matriz de valores que informa os parâmetros de distorção que a imagem está sofrendo pela câmera.

A terceira etapa do protótipo faz a junção da imagem publicitária com a imagem filmada. A figura 7 mostra a correlação dessa etapa do protótipo com o sistema tradicional.



Fonte : Adaptado Gomes (1999, p. 12)

Figura 7 - Correlação entre o sistema tradicional e o protótipo

A etapa de projeção de imagem consiste em aplicar na inserção virtual a mesma operação de projeção feita pela câmera real, desta forma, a inserção deve parecer como se realmente existisse na cena real.

2.2 RECONHECIMENTO AUTOMÁTICO DE LINHAS DE CAMPO EM ARQUIVOS DE VÍDEO PARA PUBLICIDADE VIRTUAL

O trabalho de Koser (2003) trata dos passos iniciais do processo de publicidade virtual. Segundo Koser (2003, p. 2) “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.” Foi implementado um protótipo de *software* que executa as etapas de filtragem, seleção e extração de segmento de reta.

O processo de filtragem de imagens segundo Koser (2003, p. 23) “[...] consiste em detectar pontos passíveis de estarem sobre linhas presentes na imagem e suavizar problemas presentes na imagem, como ruídos”. O método usado por Koser (2003) para a filtragem de imagens tinha como objetivo encontrar pontos de intersecção de linhas de fundo e laterais, grande área e pequena área de imagens de campos de futebol provenientes de arquivos de vídeo.

Em Koser (2003, p. 27) o processo de extração dos segmentos de reta consiste em “[...] identificar e extrair todos os segmentos de reta existentes na imagem, independente de fazerem parte do campo ou não.” Neste processo Koser (2003) conseguiu identificar, posteriormente, as retas, usando o método dos Mínimos Quadrados. A figura 8 apresenta, um exemplo de resultado final do trabalho de Koser.

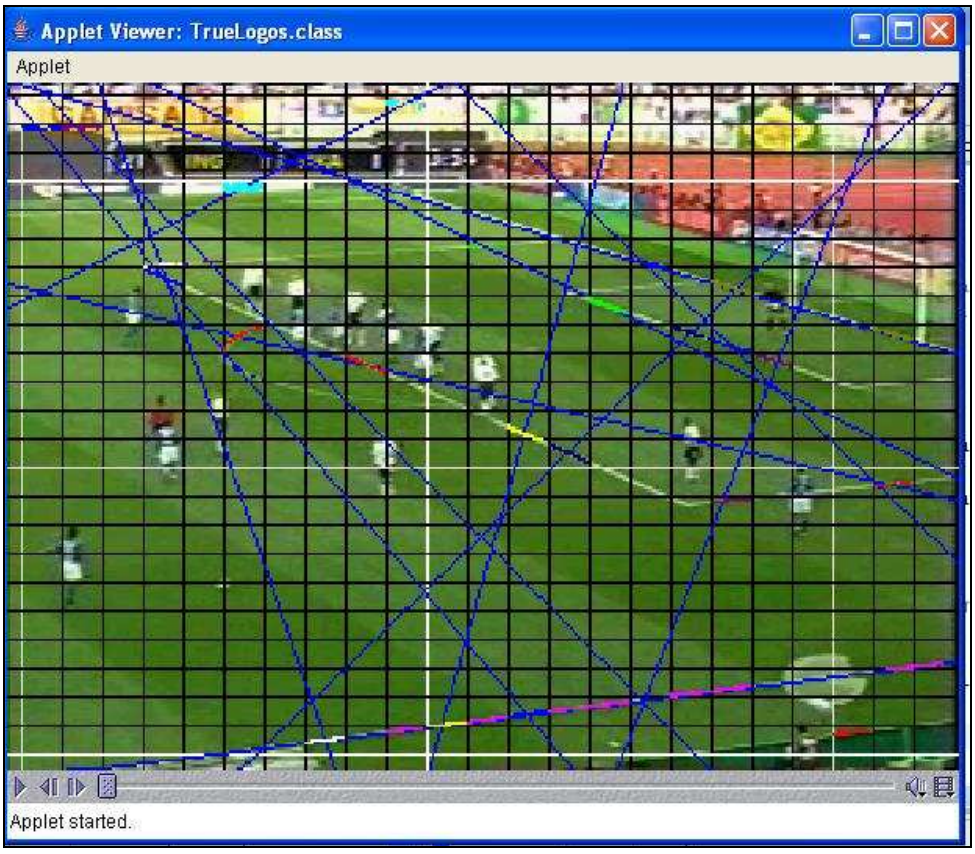


Figura 8 – Saída do protótipo de Koser

Como se pode constatar na figura 8, o trabalho de Koser (2003) realiza com grande eficiência o processo de localização de segmentos de reta. Mas a junção de segmentos que estão na mesma reta não é feita com sucesso.

2.3 UNIÃO DE SEGMENTOS DE RETA LONGOS

Esta etapa consiste em localizar retas em um conjunto de segmento de retas. Exemplo de um conjunto de segmentos de reta conforme a figura 9.

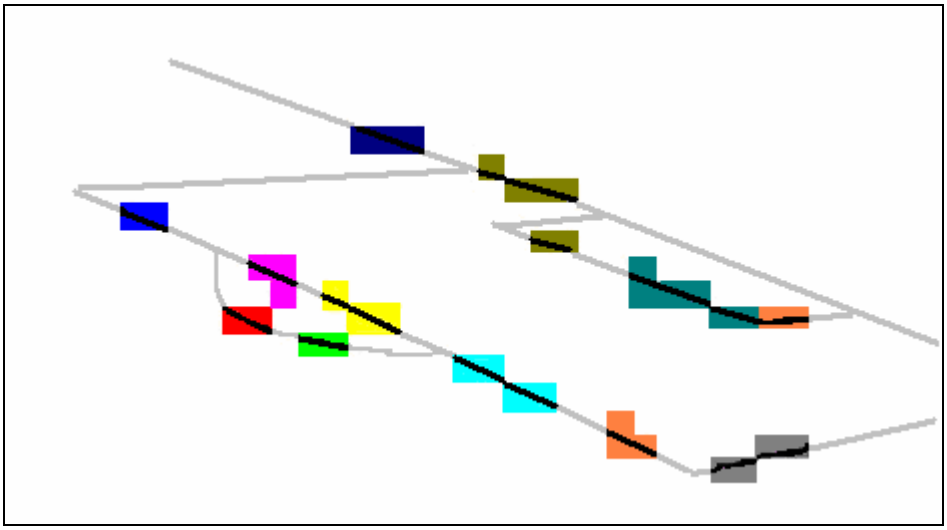


Figura 9 – Conjunto de segmentos de reta

Para a realização desta etapa, deve-se determinar a equação de reta de cada segmento de reta do conjunto. Para isto pode-se utilizar a equação de mínimos quadrados conforme quadro 1, esta equação serve para encontrar a equação da reta conforme quadro 2.

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

Fonte: Koser (2003, p. 31)

Quadro 1- Equação dos Mínimos Quadrados

$$v = au + b$$

Fonte: Koser (2003, p. 31)

Quadro 2 – Equação da reta

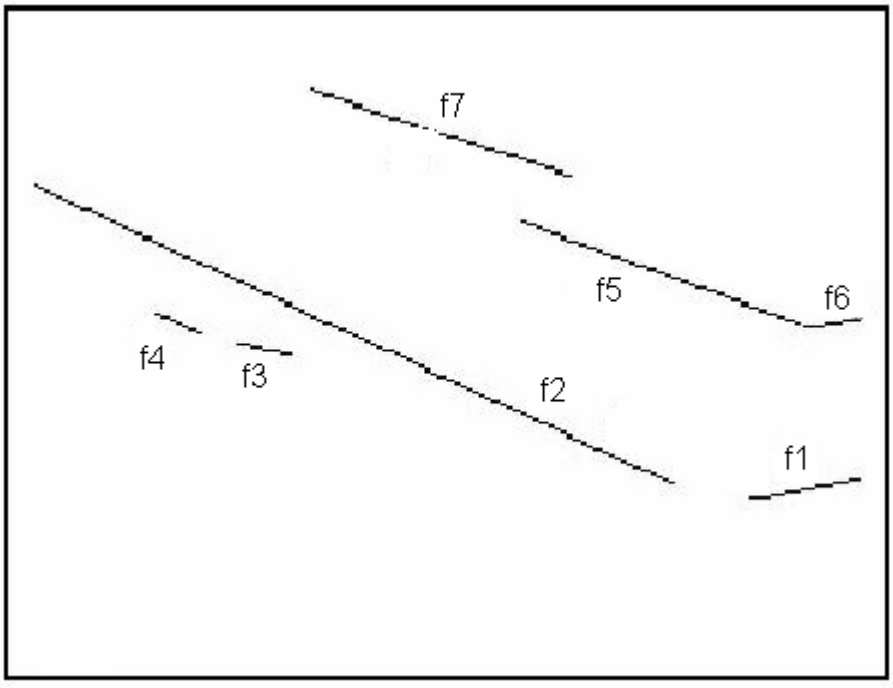
Em um campo de futebol as linhas são ortogonais entre si, porém em um vídeo existem certas particularidades que podem distorcer essa característica como aberrações,

então para isso é preciso encontrar segmentos que estejam sobre a mesma reta e possuam um ângulo de diferença pequeno entre si. Para isto é usada a equação de diferença de ângulo entre duas retas conforme quadro 3.

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

Fonte: Koser (2003, p. 31)
Quadro 3 - Diferença de ângulo entre dois segmentos de reta

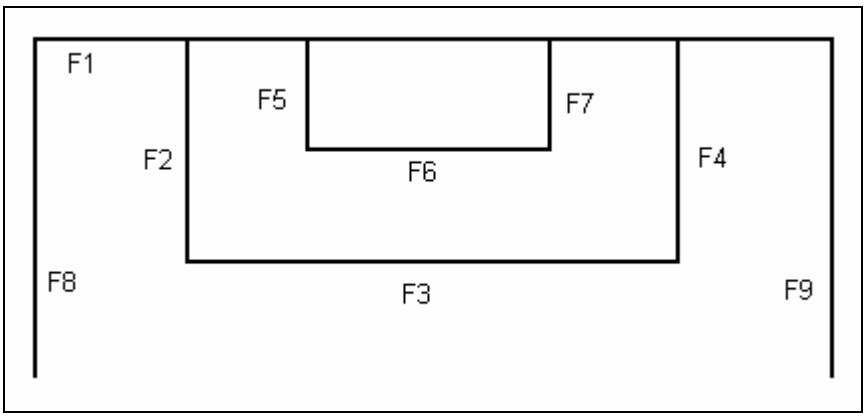
O resultado da aplicação dessas três fórmulas sobre o conjunto de segmentos da figura 9, pode ser visto na figura 10.



Fonte Szemberg (1999, p. 76)
Figura 10 - União dos segmentos de reta

2.4 RECONHECIMENTO E INTERPRETAÇÃO

Nesta etapa do trabalho, a primeira fase consiste em definir um modelo matemático do objeto cuja imagem se quer encontrar, no caso um campo de futebol, para a partir daí criar as regras matemáticas para a sua localização. Szemberg (2001, p. 66) define este método como reconhecimento baseado em modelos. Exemplo de modelo matemático figura 11.



Fonte: Szemberg (1999, p. 85)

Figura 11 – Modelo matemático ou modelo real

Neste método é usada uma estrutura de árvore chamada de “árvore de interpretação”, na qual o caminho até cada uma de suas folhas representa uma solução, a cada nível que uma folha que é visitada na árvore é feita uma interpretação para verificar a validade da possível solução. A árvore tem a profundidade máxima igual ao número de retas do modelo matemático. A figura 12 ilustra os dois primeiros níveis de uma árvore de interpretação, nesta figura, o modelo real apresenta duas linhas a menos para não sobrecarregar a figura.

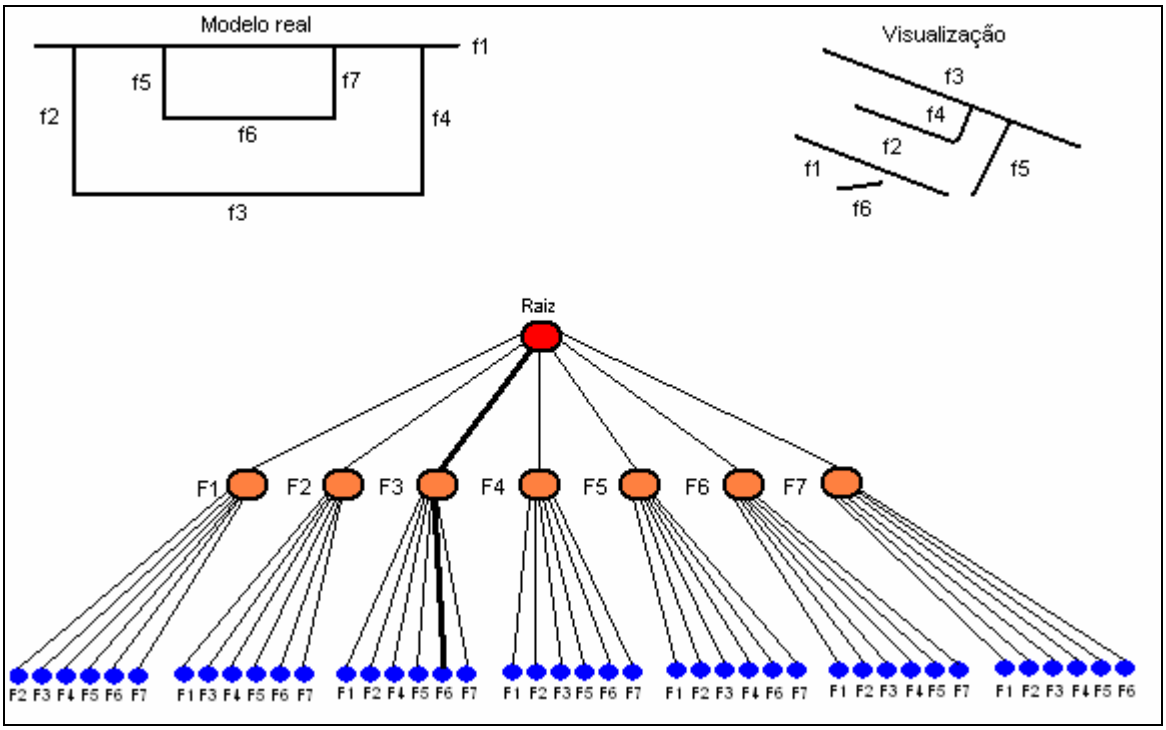


Figura 12 – Exemplo de árvore de interpretação

Características do sistema:

a) uma linha na visualização não tem, necessariamente, uma correspondente no mundo real;

b) uma linha do modelo real não tem, necessariamente, uma correspondente na visualização;

Na figura 13, seguindo o plano de execução a linha f1 da visualização se encaixou na linha f3 do modelo real, a linha f2 da visualização na linha f6 do modelo e assim por diante.

Para o exemplo da figura 13 se conseguiu a seguinte saída:

f1->f3 f2->f6 f3->f1 f4->f7 f5->f4 f6->Ø f7-> Ø

Onde fx->fy, fx linha do modelo real e fy linha da visualização.

Para que ocorra este encaixe da linha correta da visualização com a linha do modelo real é criado um conjunto de regras, convertido para regras gramaticais, conforme quadro 4.

1. Duas linhas na visualização não podem ter a mesma representação no modelo;
2. 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;
3. Todas as linhas devem estar em um mesmo semiplano determinado pela linha que representa F1;
4. Todas as linhas devem estar em um mesmo semiplano determinado pela linha que representa F8;
5. Todas as linhas devem estar em um mesmo semiplano determinado pela linha que representa F9;
6. Todas as linhas devem estar em um mesmo semiplano determinado pela linha que representa F2, exceto as representantes de F1 e F8;
7. Todas as linhas devem estar em um mesmo semiplano determinado pela linha que representa F3, exceto as representantes de F8 e F9;
8. Todas as linhas devem estar em um mesmo semiplano determinado pela linha que representa F4, exceto as representantes de F1 e F9;
9. A linha que representa F5 deve estar entre as linhas que representam F1 e F6;
10. A linha que representa F5 deve estar entre as linhas que representam F2 e F6;
11. A linha que representa F6 deve estar entre as linhas que representam F1 e F3;
12. A linha que representa F7 deve estar entre as linhas que representam F1 e F6;
13. A linha que representa F7 deve estar entre as linhas que representam F4 e F6;
14. O correspondente de F5 não pode cruzar o correspondente de F6 e vice-versa;
15. O correspondente de F7 não pode cruzar o correspondente de F6 e vice-versa.

Fonte: Szemberg (2001, p. 89)

Quadro 4 – Regras gramaticais do modelo matemático

Para fazer a validação se duas linhas estão no mesmo semiplano é usada a equação matemática, equação conforme quadro 5.

$$A=(ax,ay) \qquad B=(bx,by)$$
$$A=Pf-Pi \qquad B=Pf-Pi$$
$$\begin{cases} Ax = Pfx - Pix \\ Ay = Pfy - Piy \end{cases} \qquad \begin{cases} Bx = Pfx - Pix \\ By = Pfy - Piy \end{cases}$$

Quadro 5 – Fórmula do semi-plano

A segunda parte das validações verifica se houve intersecção entre as retas, isto consiste em verificar p1 e p2 atravessam a reta formada pelos pontos p3p4, para isso se usou a fórmula conforme quadro 6.

$$p_1 + u(p_2 - p_1) = p_3 + v(p_4 - p_3)$$

Fonte: Queiroz (2003, p.67)

Quadro 6 – Fórmula par descobrir o ponto de interseção

A igualdade da fórmula do quadro 6 dá origem a um sistema com duas equações e

duas incógnitas (*u* e *v*), conforme quadro 7.

$$\begin{cases} x_1 + u(x_2 - x_1) = x_3 + v(x_4 - x_3) \\ y_1 + u(y_2 - y_1) = y_3 + v(y_4 - y_3) \end{cases}$$

Fonte: Queiroz (2003, p.67)

Quadro 7 - Sistema para resolver o problema de intersecção de retas

Desenvolvendo o sistema do quadro 7, chega-se ao valor das duas incógnitas, conforme o quadro 8.

$$u = \frac{(x_4 - x_3)(y_1 - y_3) - (y_4 - y_3)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)}$$

$$v = \frac{(x_2 - x_1)(y_1 - y_3) - (y_2 - y_1)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)}$$

Fonte: Queiroz (2003, p.67)

Quadro 8 – Sistema de intersecção valor das incógnitas

Calculados os parâmetros u e v, pode-se determinar o ponto de interseção, conforme quadro 9.

$$\begin{cases} x_{intersecção} = x_1 + u(x_2 - x_1) \\ y_{intersecção} = y_1 + u(y_2 - y_1) \end{cases} \text{ ou } \begin{cases} x_{intersecção} = x_3 + v(x_4 - x_3) \\ x_{intersecção} = y_3 + v(y_4 - y_3 \end{cases}$$

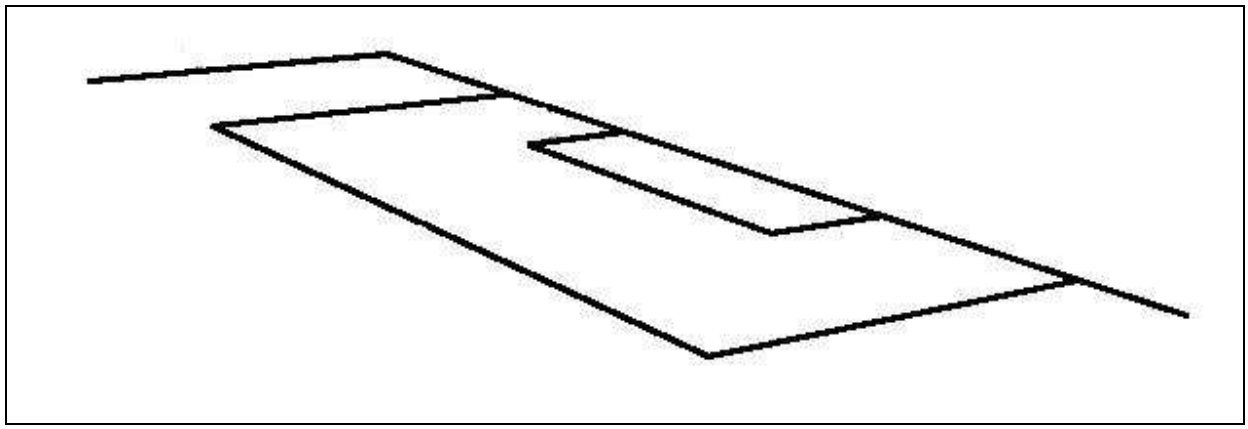
Fonte: Queiroz (2003, p.68)

Quadro 9 – Achando os pontos de intersecção

As expressões *u* e *v* possuem interpretações importantes. Os denominadores são os mesmos, portanto em uma implementação computacional devem ser calculados uma única vez. Se o denominador for igual a zero as linhas são paralelas. Se além dos denominadores os numeradores de ambos os parâmetros também forem iguais a zero, então as linhas são coincidentes.

Também é realizado o teste para saber se as retas são paralelas ou não, já que duas

retas que são paralelas no modelo também devem ser na visualização. Na visualização da imagem encontra-se um problema devido à distorção provocada na imagem pela lente da câmera, as retas aparecem em uma projeção perspectiva, conforme figura 13.



Fonte: Szenberg (2001, p. 85)

Figura 13 – Visualização de um campo de futebol com distorção perspectiva

Para solucionar este problema foi usado na formula do paralelismo o efeito *n* que é o fator que representa a distorção da imagem, conforme quadro 10.

$$\frac{\left|(S_{u2}-S_{u1})(T_{u2}-T_{u1})+N^2(S_{v2}-S_{v1})(T_{v2}-T_{v1})\right|}{\sqrt{(S_{u2}-S_{u1})^2+N^2(S_{v2}-S_{v1})^2}\sqrt{(T_{u2}-T_{u1})^2+N^2(T_{v2}-T_{v1})^2}} \geq \rho$$

Fonte: Szenberg (2001, p. 90)

Quadro 10 - Equação do paralelismo, com fator distorção

O resultado desta formula será *p*, onde *p* representa a tolerância ao paralelismo, já que *p* é o valor do co-seno entre as duas retas testadas.

Para este trabalho, o valor usado para *n* foi 2, para suavizar a abertura dos ângulos na imagem. O valor *n* trata-se de uma constante que, dependendo da distorção da imagem pode exigir correção do valor.

Neste trabalho, verificou-se que, dificilmente as retas do modelo real para as retas da imagem têm uma diferença maior do que 36°, por isso o valor usado como tolerância ao paralelismo usado foi 0,81.

Quando se encontrou mais de uma solução válida, foram usados alguns critérios de desempate, estes critérios estão representados no quadro11.

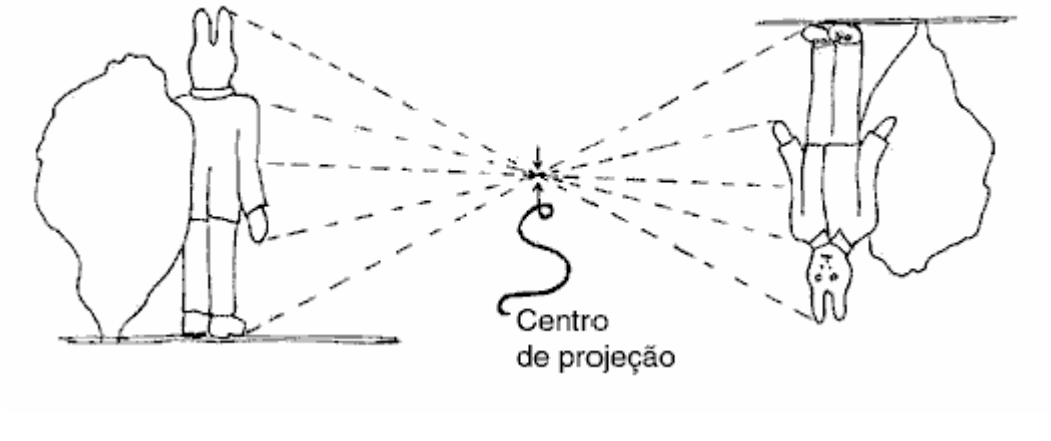
- **vence a solução com o maior número de linhas que tenham representações;**
- **vence a solução que tenha a maior soma dos comprimentos das linhas que tenham**
- **representações;**
- **vence a solução que apresente a maior linha com representação da linha F0 do**
- **modelo real;**
- **combinação dos exemplos acima.**

Fonte: Szenberg (2001, p. 92)

Quadro 11 – Regras de desempate

2.5 CÂMERAS DE VÍDEO

Este trabalho baseia-se no modelo de câmera de vídeo *pinhole*, que segundo Szemberg (2001, p. 97) o seu funcionamento se baseia nos raios que partem do objeto no mundo tridimensional e passam por um orifício formando uma imagem invertida. Conforme figura 14.



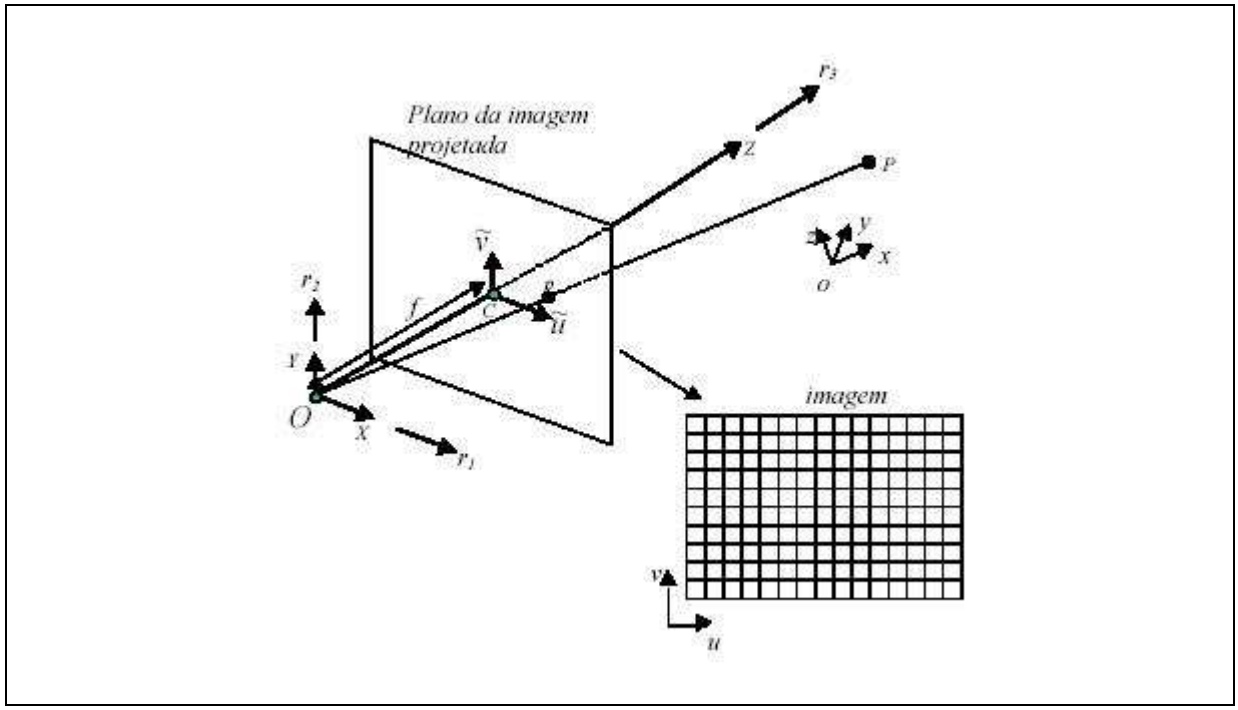
Fonte: Szenberg (2001, p. 98)

Figura 14 – Projeção através de um ponto

Segundo Szemberg (2001, p. 98) em computação gráfica este modelo é modificado passando a imagem a se formar entre o objeto tridimensional e o centro de projeção, a imagem fica a uma distância focal denotada por *f*.

Em um sistema tridimensional as coordenadas são dadas por OXYZ, com o centro

óptico da imagem dada por Z, onde este eixo, neste trabalho, é igualado a 0, surgindo um sistema formado com os eixos X e Y.Conforme figura 15.



Fonte: Szenberg (2001, p. 98

Figura 15 – Modelo de projeção de imagem da câmera “pinhole”

2.6 CALIBRAGEM DE CÂMERAS

No caso de se tentar tratar de elementos que estejam em uma vista em perspectiva, é necessário descobrir a câmera que gerou a imagem em estudo. Segundo Szemberg (2001, p. 104) “Uma câmera é composta, ou pode ser modelada, por parâmetros intrínsecos, compostos pela geometria interna e características ópticas, e extrínsecos, dados pela sua localização, direção de visualização e inclinação”. Com estas informações da câmera, pode-se realizar uma modelagem da cena.

Existem alguns métodos que tratam deste problema. Segundo Szemberg (2001, p. 104) “Um método clássico e conhecido na literatura, chamado Método de Tsai, e outro utilizado pelo sistema Juiz Virtual.”

O método de Tsai, segundo Szemberg (2001, p. 15) baseia-se

> [...] no conceito de linhas e pontos de fuga e usando um paralelepípedo regular como objeto de calibração. O modelo foi desenvolvido a partir da equação de colinearidade, encontrando-se expressões que relacionam os pontos de fuga com os parâmetros intrínsecos e extrínsecos.

Esta técnica é usada para imagens onde há uma cena arquitetônica urbana isto é, não se aplica a imagens planares.

O método do Juiz Virtual, segundo Szemberg (2001, p.106), tem o propósito de

> dada uma imagem proveniente de uma transmissão de televisão de uma partida de futebol, encontrar a câmera responsável por essa imagem e a partir disto reconstruir a cena 3D, com o objetivo de mover a câmera para outro ponto e redirecioná-la, a fim de poder observar e analisar o mesmo lance a partir de outro ponto de vista.

Estes métodos procuram descobrir os parâmetros de definição da câmera baseando-se em pontos conhecidos na imagem.

A diferença entre o método de Tsai e o método do Juiz Virtual, é que o método de Tsai busca descobrir a homografia da imagem, enquanto o método do Juiz Virtual busca a câmera e com este tem-se a vantagem de reconstruir elementos que estão fora do plano da imagem, imagem esta que gerou os pontos para a posterior localização da câmera.

Para este trabalho, há real necessidade de se encontrar a câmera, e com os pontos descobertos na imagem se chegar às coordenadas para a inserção da publicidade virtual no vídeo.

2.7 ACOMPANHAMENTO DE CENAS COM CALIBRAÇÃO AUTOMÁTICA DE CÂMERAS

Segundo Szemberg (2001, p. 3) “Nesta tese, é apresentado um algoritmo para recuperar, em tempo real e sem utilizar qualquer informação adicional, a posição e os

parâmetros da câmera em uma seqüência de imagens contendo a visualização de modelos conhecidos.”

O problema resolvido por Szemberg (2001, p. 26) é “Dada uma seqüência de imagens que apresentam a visualização, total ou parcial, de um determinado modelo, acompanhar este modelo, calibrando as câmeras para cada imagem de forma automática, a fim de sobrepor objetos virtuais.”

Szemberg (2001) procurou com imagens de entrada de um campo de futebol conseguir localizar a câmera que gerou essas imagens.

O seu trabalho foi desenvolvido para chegar no objetivo descrito anteriormente da seguinte forma: primeiro ele desenvolveu um método para extração de segmentos longos de reta. O passo seguinte foi criar um modelo matemático de um campo de futebol. Com as linhas extraídas do primeiro passo, ele tentou localizar essas linhas no seu modelo matemático. Mapeando essas linhas para o modelo matemático e localizando as linhas no modelo matemático. Essas linhas são passadas como parâmetros para a outra etapa do trabalho, que consiste em encontrar a câmera responsável pela visualização do modelo.

Esse trabalho apresentou bons resultados quando a seqüência de imagens mostra a maior parte das linhas do modelo matemático com razoável nitidez. Os tempos de processamento neste caso se mostraram viáveis para serem usados numa transmissão ao vivo.

Algumas situações em que o algoritmo de Szemberg (2001) não se mostrou viável, foi quando existia sombra parcial sobre as linhas do campo e quando havia menos de quatro linhas do modelo matemático visíveis na imagem.

3 DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo tem por objetivo apresentar as etapas envolvidas no desenvolvimento do protótipo proposto por este trabalho. Serão abordadas as seguintes etapas: requisitos do protótipo, especificação, implementação e, finalmente, serão apresentados os resultados obtidos.

3.1 REQUISITOS PRINCIPAIS DO PROTÓTIPO

a) sistema deve permitir a entrada de um arquivo de vídeo, com cenas de um campo de futebol e a partir deste vídeo ele deverá extrair retas da imagem do campo de futebol. Estas retas devem ser unidas em segmentos longos de reta (requisito funcional);

b) o sistema deverá ser capaz de mapear estas retas encontradas na imagem para um modelo matemático ou modelo real da imagem do campo de futebol. Para então encontrar um conjunto de pontos cuja localização é conhecida tanto na imagem como no modelo matemático ou modelo real (requisito funcional).

c) o sistema deverá extrair as propriedades da câmera que gera o arquivo de vídeo para realizar uma calibração desta câmera. Com isso, realizar a inserção da imagem de publicidade virtual no vídeo que recebeu de entrada (requisito funicional);

d) ser programado em Java (requisito não funcional);

e) utilizar do Java o *Java Media framework* (JMF) (requisito não funcional).

3.2 ESPECIFICAÇÃO

Para realizar a especificação do protótipo foi utilizada a linguagem UML, descrita em Furlan (1998). Como diagramas da especificação foram usados os diagramas de caso de uso, de classe e de seqüência do protótipo. Como os objetivos do trabalho não foram concluídos em sua totalidade, serão apresentados os diagramas das partes que foram efetivamente implementadas pelo autor.

3.2.1 Diagrama de casos de uso

A Figura 16 apresenta o diagrama de casos de uso, gerado na especificação do protótipo.

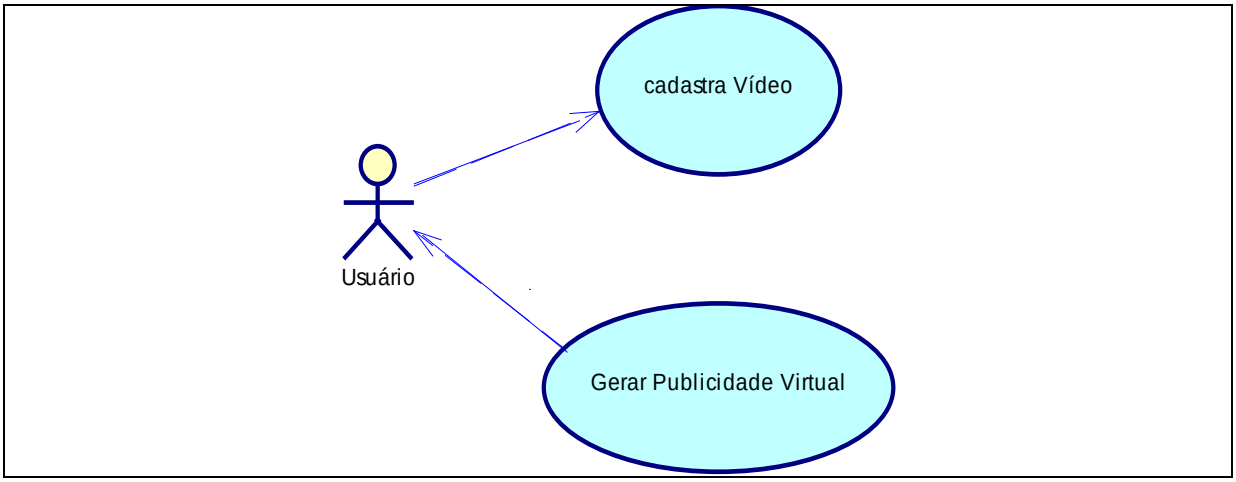


Figura 16 – Diagrama de casos de uso

O diagrama de caso de uso é bastante simples, pelo fato do usuário não ter muita interação com o protótipo. A única tarefa necessária do usuário no sistema, é informar um arquivo de vídeo com cenas de um jogo de futebol, e o sistema terá como saída o reconhecimento e identificação das cenas do campo de futebol.

3.3 IMPLEMENTAÇÃO

O usuário deve informar um vídeo de entrada, contendo cenas de um campo de futebol e a saída do sistema deve ser o vídeo com a publicidade virtual.

3.3.1 Diagrama de classes

Para a geração destes diagramas de classe foi usada a ferramenta PowerDesigner versão 9.5. Ela trata métodos e atributos públicos com o sinal "+" na sua frente, e os métodos e atributos privados com o sinal de "-" na sua frente.

Na figura 17 é mostrado o diagrama com a relação entre as classes dessa implementação. Para essa implementação foi usado o conceito de *package*, onde todas as classes mostradas fazem parte da package Arvore.

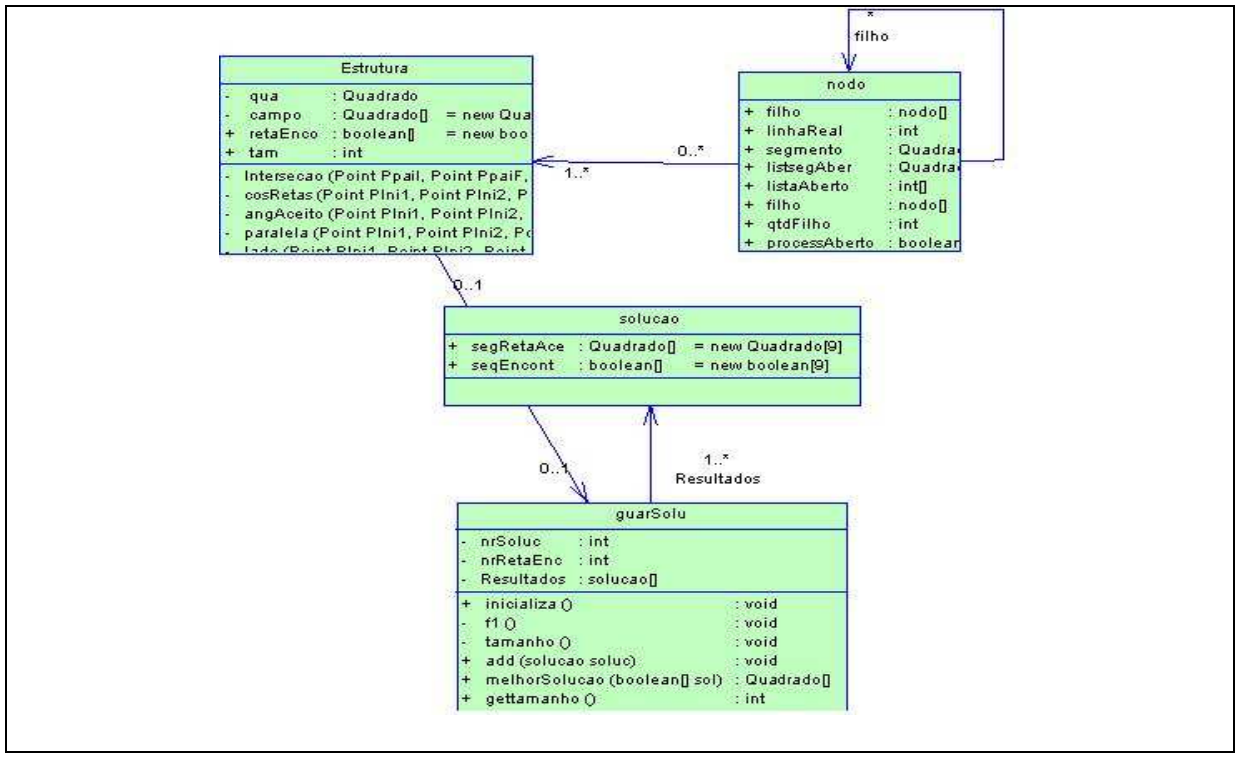


Figura 17 – Interação entre as classes da package Arvore

A figura 18 mostra mais detalhadamente a classe Estrutura, que se trata da classe principal desta especificação.

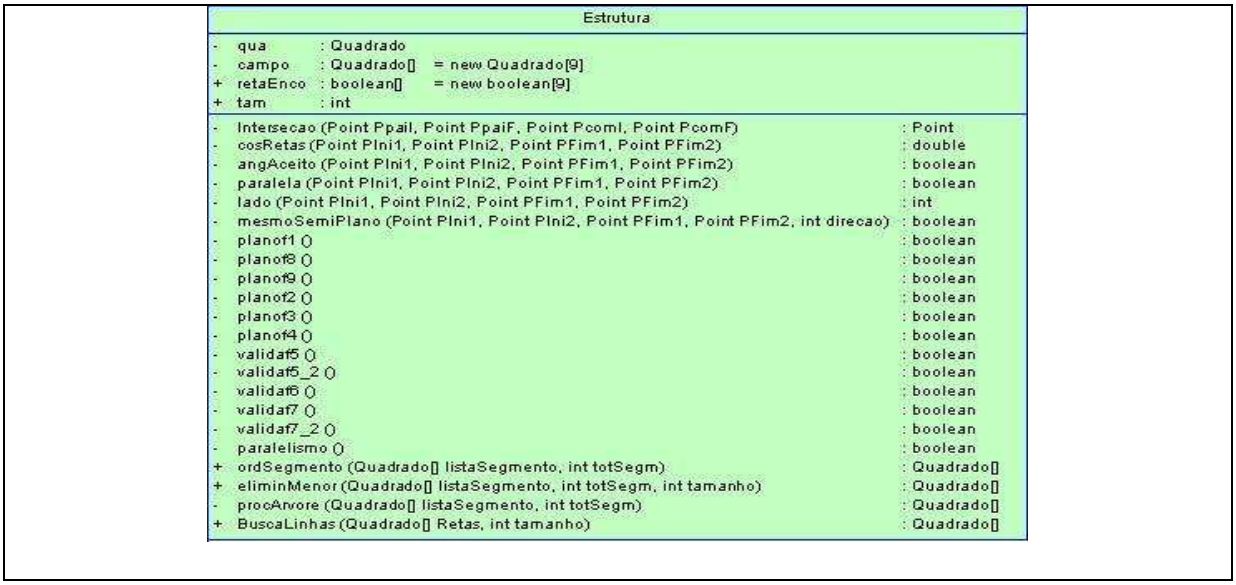


Figura 18 – Classe Estrutura

A classe estrutura é responsável por organizar todo o processo de mapeamento das retas encontradas na imagem para as retas do modelo real ou matemático.

O método principal a ser chamado pelo programa desta classe é o método BuscaLinhas, que recebe o conjunto de retas encontradas na etapa anterior do trabalho. Esta classe corresponde aos conceitos apresentados na sessão 2.4.

Para melhorar o desempenho, o mapeamento dessas retas da imagem para o modelo matemático foram criados mais dois métodos. O método eliminaMenor, que retira do conjunto de segmentos retas muito pequenas que prejudicariam o processo de mapeamento e o método ordSegmento, que ordena os segmentos do maior para o menor em relação ao tamanho já que se provou que os segmentos de reta com maior tamanho eram candidatos com maior possibilidade de serem retas no modelo matemático

Os métodos planof1, planof8, planof9, planof2, planof3, planof4, validaf5, validaf5_2, validaf6, validaf7, validaf7_2 foram criados para validar as regras do quadro 4. Dentro desses métodos são implementados os cálculos de intersecção, que retorna o ponto de intersecção entre duas retas, O método cosRetas que retorna o co-seno entre duas retas, o método

angAceito, valida o ângulo entre as duas retas, levando em consideração a visão em projeção do campo de acordo com a figura 14. O método mesmoSemiPlano valida se as retas estão no mesmo semiplano, levando em consideração o lado que se encontra a reta em observação usando o método lado. O método paralelismo valida as retas que são paralelas no modelo real ou matemático devendo ser na imagem observada.

A figura 19 mostra a classe nodo responsável por constituir as folhas da árvore discutida na sessão 2.4.

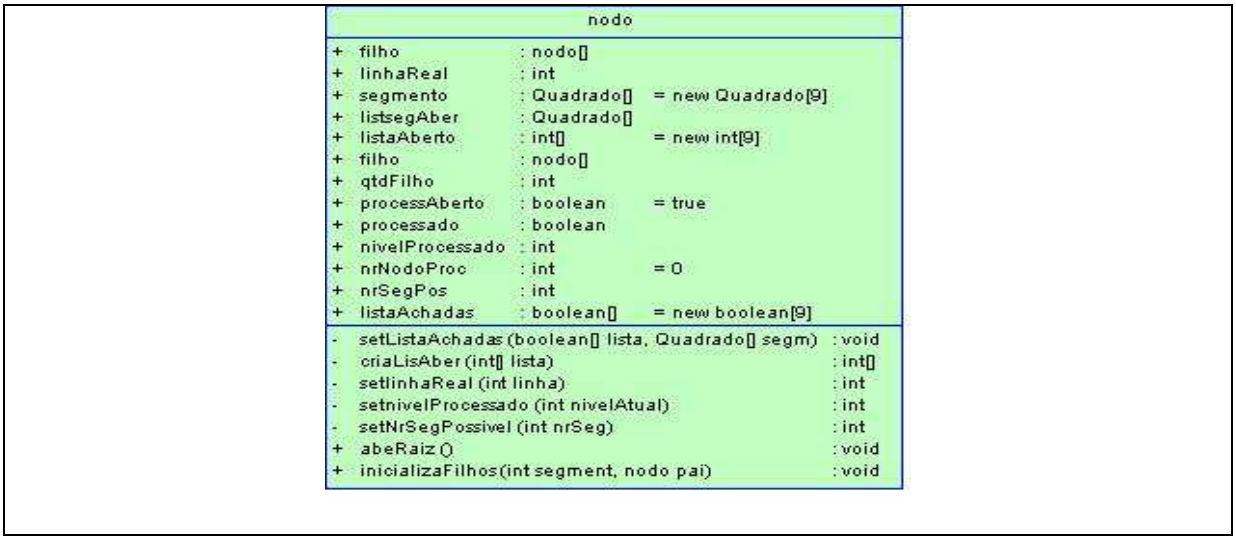


Figura 19 – Classe nodo

Os métodos desta classe são em geral métodos de inicialização para controle dos atributos internos da classe. Os atributos desta classe são todos públicos porque são manipulados pela classe estrutura. A figura 20 mostra a classe solução e a guarda-solução.

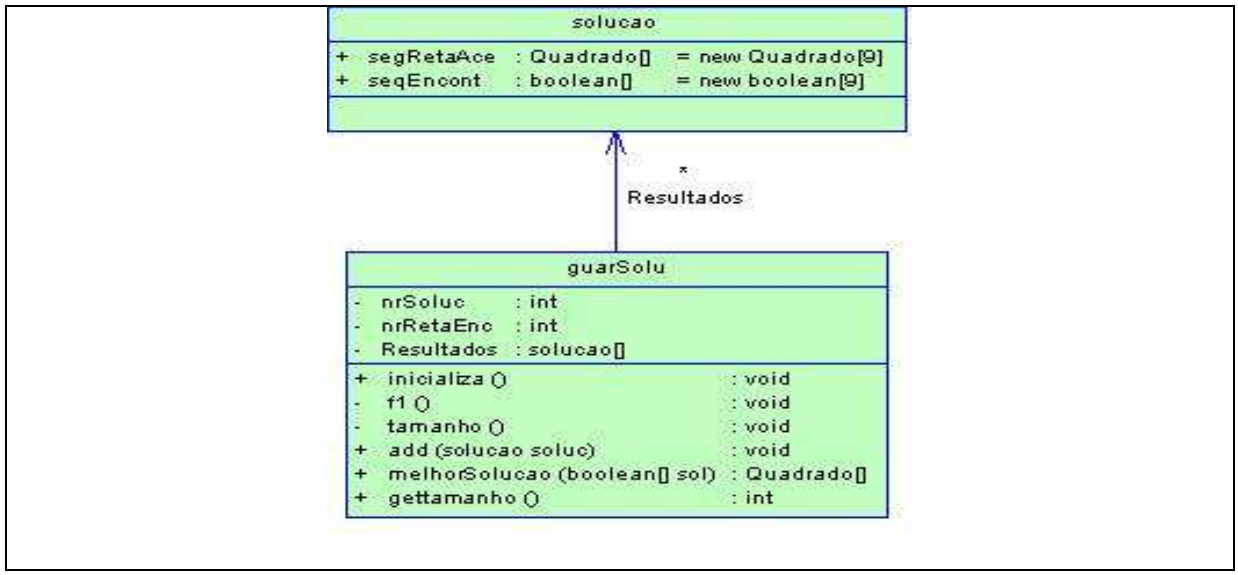


Figura 20 – Classes solução e guarda-solução

A classe guarda-solução é responsável por armazenar as soluções válidas e em casos de mais de uma solução válida satisfazer as regras do quadro 11. Sendo que ela guarda sempre só as soluções que tenham a mesma quantidade de retas encontradas no modelo matemático, a partir do momento que uma nova solução é encontrada com mais retas mapeadas ela descarta todas as outras soluções.

O método tamanho é usado no desempate de soluções, levando em consideração a solução com maior tamanho na soma dos segmentos de reta encontrados. O método f1 também usado no desempate de soluções leva em consideração a solução com o maior segmento de reta que corresponde à reta f1 do modelo real.

3.3.2 Diagrama de seqüencia

Na figura 21, é apresentado o diagrama de seqüência das classes implementadas. Com uma visão macro de como o sistema funciona, a cada solução que é encontrada.

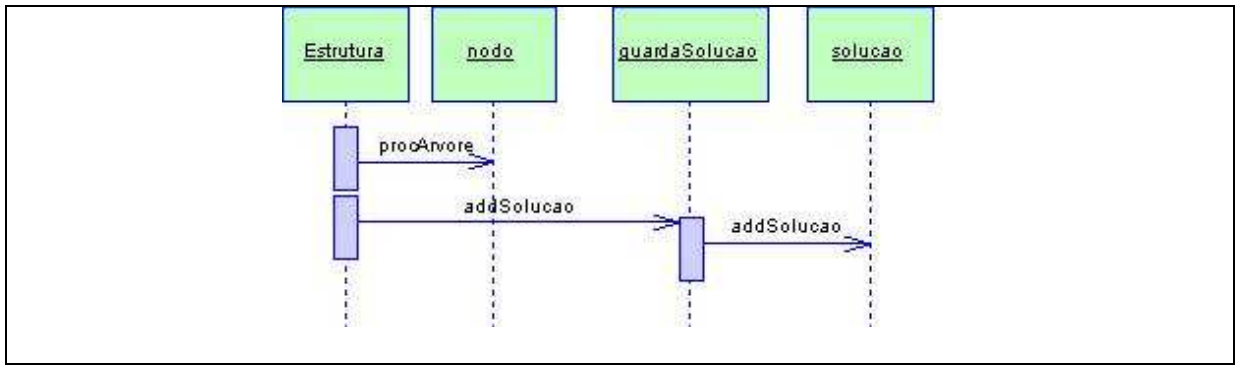


Figura 21 – Diagrama de seqüência

No método procArvore é criado a árvore de interpretação. Esta árvore usa a classe nodo para gerir as folhas da árvore. Ao final de cada novo segmento válido encontrado para o nodo da árvore é chamada a classe guardaSolucao, que decide se deve ou não adicionar esta solução ao conjunto de soluções válidas.

3.4 IMPLEMENTAÇÃO

Este item apresenta considerações sobre a implementação do protótipo, como técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.4.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento deste trabalho foram utilizadas a linguagem Java e a sua API Java Media FrameWork ou JMF. O JMF foi utilizado porque é distribuído gratuitamente Sun (2005), é uma API específica para o tratamento de vídeo e música, suportando vários formatos de vídeo e música.

Para esta API funcionar, é necessário que seja instalado antes, a máquina virtual Java, Java Second Edition ou J2SE.

O ambiente utilizado para desenvolvimento deste protótipo foi o Eclipse Plataform

versão 3.0.1 – informações sobre este ambiente são encontradas em Eclipse (2005).

O trabalho tem a representação no plano das coordenadas *x* e *y* da seguinte forma, conforme demonstrado na figura 22.

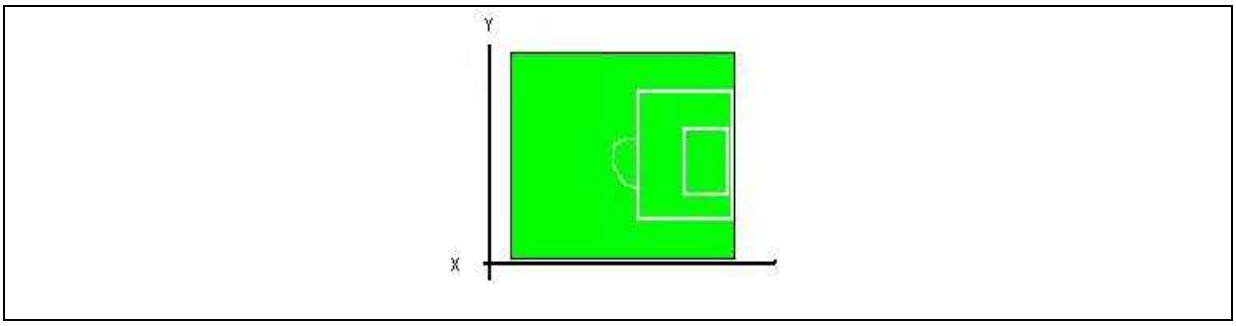


Figura 22 – Sistema de Coordenadas

A primeira parte a ser resolvida no protótipo foi acertar a junção de segmentos de reta em segmentos longos de reta.

Este problema pode ser visto na figura 23.

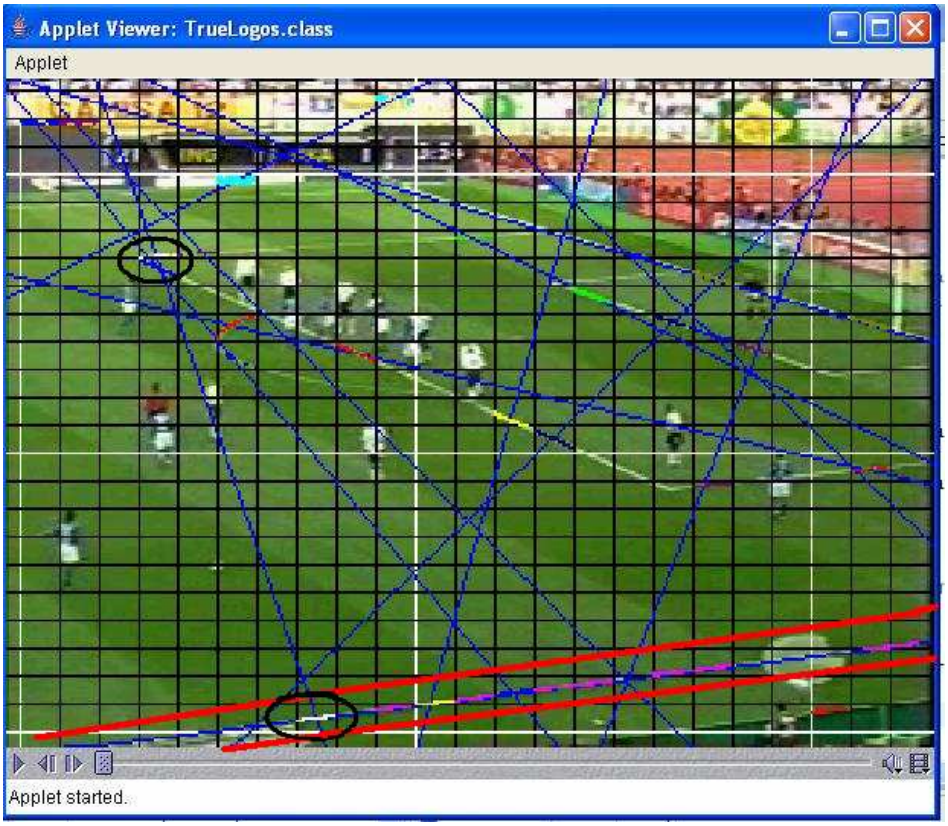


Figura 23 - Erro resolvido

Na figura 23 o erro pode ser percebido da seguinte forma, nos círculos preto estão os

segmentos que estão sendo unidos, isto esta errado, porque ele deveria unir os segmentos no sentindo do retângulo vermelho colocado na figura. As retas em azul na figura 23 são demonstra como os segmentos de reta estão sendo unidos.

Isto foi resolvido, acertando o código do quadro 12.

```
while ((i < matQua.segRetAtu) && (qua.segReta == -1)) {
      // Compara para saber se o quadrado pertence à célula
      if (matQua. Compara Células (qua, matQua.segRetas[i], true)) {

         if (i == 0) {
            qua.cxX = -1;
            qua.cxY = -1;
            matQua.segRetas[i].cxX = -1;
            matQua.segRetas[i].cxY = -1;
         }
         matQua.comparaCélulas(qua, matQua.segRetas[i], true);
         // Se for o mesmo Segmento // Adiciona nos
         // quadrados que fazem parte deste segmento de reta
         matQua.segRetas[i].AddQuad(qua);
         // Diz que o quadrado pertence ao segmento...
         qua.segReta = i;
         }
      i++;
  }
 // Se o quadrado não pertence a nenhum segmento de reta,
 // o quadrado é uma nova linha
 if (qua.segReta == -1) {
    // Adiciona um novo segmento
    matQua.segRetAtu++;
    // Adiciona o quadrado no segmento recem inserido
    matQua.segRetas[matQua.segRetAtu].AddQuad(qua);
    //Determina as linhas para saber o tamanho da reta
    matQua.segRetas[matQua.segRetAtu].pIni
          .setLocation(qua.pIni);
    matQua.segRetas[matQua.segRetAtu].pFim
          .setLocation(qua.pFim);
    matQua.segRetas[matQua.segRetAtu].segReta = -1;
    qua.segReta = matQua.segRetAtu;
```

Quadro 12 – Código ajustado

O que acontecia no protótipo anterior é que na, inicialização dos pontos de início do novo segmento de reta, eles eram sempre iniciados com zero, ocorrendo que nenhum ou poucos segmentos eram adicionados um aos outros e os que eram, geralmente não correspondiam ao segmento correto a ser ajustado, tendo o sistema a saída conforme figura 9. Com a correção deste problema o protótipo começou ter a saída conforme figura 24.

Figura 24 – Saída do protótipo com a correção da junção dos segmentos de reta

A etapa conseqüente à anterior foi realizar o processo interpretação destas retas. Para isto foi adicionado ao programa principal a chamada para as rotinas desenvolvidas que realizam este processo de interpretação conforme quadro 11.

```
//faz arvore de decisão
//começa a processar a busca
//de linhas do modelo físico
arv = new Estrutura();
segRetas    = new Quadrado[matQua.segRetAtu];
//atribições
 segRetas   = matQua.segRetas;
 arv.tam    = matQua.segRetAtu;
 //processamento e atribui
 segRetas  = arv.BuscaLinhas(segRetas,arv.tam);
```

Quadro 13 – Código que realiza a chamada para as rotinas implementadas neste protótipo.

Neste momento começa a ser criada a arvore de interpretação, conforme quadro 12.

```
nodo processado = new nodo( );
nodo raiz       = new nodo( );
guarSolu resSol = new guarSolu( );
//inicializa solução
resSol.inicializa( );
raiz.nrSegPos = totSegm + 1;
//inicializa a nodo raiz com todos as linhas a percorrer
raiz.abeRaiz( );
//cria a lista de possíveis filhos
//raiz.listsegAber = eliminMenor (listaSegmento,totSegm,17);
//raiz.nrSegPos = totSegm;
raiz.listsegAber = ordSegmento(listaSegmento,totSegm);
//deve ser a última inicialização
raiz.inicializaFilhos(-1, raiz);
//processado recebe o primeiro filho
processado = raiz.filho[raiz.nrNodoProc];
int cont = 0;
while (raiz.processAberto) {
```

Quadro 14 – Código do inicio da arvore de interpretação

O processo de checagem das regras definidas do quadro 4 e feito pelo seguinte trecho de código, do quadro 13.

```
while ((i <  processado.nrSegPos) && (!processado.processado)) {
   int posSegmento;
   posSegmento = -1;
   campo = processado.segmento;
   campo[processado.linhaReal - 1] = processado.listsegAber[i];
   retaEnco = processado.listaAchadas;
   retaEnco[processado.linhaReal - 1] = true;
   processado.processado = false;
   //começa a processar as validações
   processado.processado = true;
   if (planof1()) {
      if (planof8()) {
         if (planof9()) {
            if (planof2()) {
               if (planof3()) {
                  if (planof4()) {
                     if (validaf5()) {
                        if (validaf5_2()) {
                           if (validaf6()) {
                              if (validaf7()) {
                                 if (validaf7_2()) {
                                    processado.processado = true;
                                 }
                              }
                           }
                        }
                     }
                  }
               }
            }
         }
      }
   }
```

Quadro 15 – Checagem das regras do quadro 4

As regras na sua constituição não se alteram muito na sua construção no quadro 16, tem-se o exemplo de como um destes métodos foi criado.

```
//verifica se a reta um foi encontrada
if (retaEnco[0]) {
    for (int i = 1; i < 9; i++) {
        if (retaEnco[i]) {
            if (!mesmoSemiPlano(campo[0].pIni, campo[i].pIni,
                campo[0].pFim, campo[i].pFim, 1)) {
                    return (false);
             }
          }
       }
   }
   //se ainda nao foi diz que a validação eh verdadeira
  return (true);
```

Quadro 16 – Exemplo de código de uma regra do quadro 4

Quando ocorrem as validações da regra que foram definidas no quadro 4, o sistema passa a solução para a classe guarSol conforme código do quadro 15.

```
if (processado.processado) {
      //o zero corresponde no vetor a linha 1
      processado.listaAchadas[processado.linhaReal - 1] = true;
      processado.segmento[processado.linhaReal - 1] = processado.listsegAber[i];
     //cria a nova lista de filhos pra baixar um nivel
     processado.inicializaFilhos(i, processado);
     //se encontrou uma solução pega o próximo no do mais abaixo
     //tem guardar esta solução
     solucao sol = new solucao( );
     sol.segRetaAce = processado.segmento;
     sol.seqEncont = processado.listaAchadas;
     resSol.add(sol);
}
```

Quadro 17 – Código onde é feito o processo em que se guardam as soluções

Realizado todo o processo tem-se a saída do sistema, conforme a figura 25.

Figura 25 – Saída do protótipo após interpretação das retas

Nesta saída pode-se notar que para esta imagem o sistema conseguiu obter bastante êxito. Ao identificar as retas reais do sistema matemático, descartando as que não representavam retas no modelo real. No caso desta imagem, as retas que foram eliminadas eram as que apareciam na meia lua da grande área, e na parte superior da imagem. Bem como as retas que apareciam junto à arquibancada.

O passo seguinte consiste em identificar os pontos de intersecção do conjunto de segmentos de reta, isso é feito através do código do quadro 16.

```
/**retorna o ponto de intersecção entre duas retas
 *
 * @param PpaiI
 * @param PpaiF
 * @param PcomI
 * @param PcomF
 * @return
 */
private Point Intersecao(Point PpaiI, Point PpaiF, Point PcomI, Point PcomF) {
        float parte1;
        float parte2;
        float uCons;
        int x = 0;
        int y = 0;
        Point pInters = new Point( );
        //primeira parte da equação
        parte1 = (PcomF.x - PcomI.x) * (PpaiI.y - PcomI.y)
                        - (PcomF.y - PcomI.y) * (PpaiI.x - PcomI.x);
        //segunda parte da equação
        parte2 = (PcomF.y - PcomI.y) * (PpaiF.x - PpaiI.x)
                        - (PcomF.x - PcomI.x) * (PpaiF.y - PpaiI.y);
        //acha um valor para cálculo dos valores x e y da intersecção
        //existem duas possibilidades de dar erro aqui
        //quando as retas forem paralelas perfeitas
        //ou quando for testado dois segmentos de reta
        //que pertecem à mesma reta
        if (parte2 == 0) { //retas paralelas
                return (PcomI);
        } else {
                uCons = parte1 / parte2;
        }
        //calculo do x da intersecção
        pInters.x = Math.round(PpaiI.x + uCons * (PpaiF.x - PpaiI.x));
        //calcula o y da intersecção
        pInters.y = Math.round(PpaiI.y + uCons * (PpaiF.y - PpaiI.y));
        //retorna o ponto da intersecção
        return (pInters);
}
```

Quadro 18 –Código para calcular o ponto de intersecção

Com base na imagem da figura 24, é possível localizar os seguintes pontos de intersecção conforme figura 26.

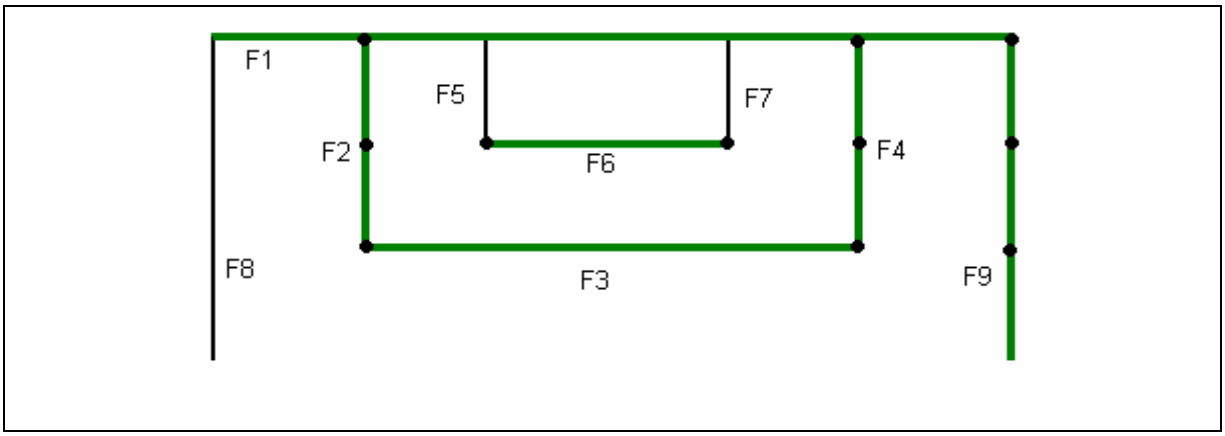


Figura 26 – Pontos de Intersecção

Na figura 25, as retas que estão em verde representam as retas encontradas na imagem 24, os pontos em preto são os pontos possíveis de serem calculados com base nestas retas.

Note que além dos pontos de intersecção natural das retas, que seriam em um exemplo a junção da reta f3 com f4, usa-se a idéia de prolongar a reta f6 para encontrar o seu ponto de intersecção nas retas f2, f4 e f9. A mesma idéia é usada com a reta f3.

3.4.2 Operacionalidade da implementação

O protótipo foi implementado como Applet, que é um programa específico para ser chamado por um navegador internet a partir de uma página HTML. Desta forma, foi abstraído da implementação toda uma interface e passagem de parâmetros para o protótipo.

A versão anterior do protótipo tinha uma grande quantidade de passagem de parâmetros, o que nesta versão foi abstraída, isto foi substituído por constantes mais genéricas; dentro do sistema no quadro 10 tem-se um exemplo de código HTML para chamar o Applet.

```
<html>
<body>
<applet code=TrueLogos.class width="200" height="200" >
</applet>
</body>
</html>
```

Quadro 19 – Exemplo de um código HTML que chama o protótipo

3.5 RESULTADOS E DISCUSSÃO

Um dos maiores problemas encontrados na realização deste trabalho foi o entendimento do código escrito por Koser (2003), pelo alto grau de complexidade, já que não se trata de um problema trivial, somado ao pouco uso de comentários dentro dos fontes. O que

só foi possível de ser resolvido com a ajuda do próprio Koser.

Esta etapa era totalmente necessária, porque não existia a possibilidade de implementar a árvore de interpretação sem antes conseguir um conjunto de retas bastante próximo da realidade.

Na etapa da árvore de interpretação, foi extremamente difícil a realização de testes com o uso da ferramenta de *debug*, pois para uma única imagem as possibilidades de encaixamento de retas são milhares. As regras matemáticas tiveram que ser validadas primeiro, em um modelo matemático. Já no protótipo, o problema de validar a entrada da imagem com a saída é que a ferramenta de visualização do vídeo (JMF) não permite visualizar o vídeo *frame* a *frame*, o que existe é uma visualização cena a cena, e para a ferramenta a cena corresponde acerca de quatro *frames*.

Isto causou uma certa confusão no início da realização dos testes por se desconhecer esta característica da ferramenta.

4 CONCLUSÕES

A idéia inicial do trabalho era realizar o processo completo de publicidade virtual, já que o início do problema já havia sido iniciado por Koser (2003) com grande êxito, mas as fases subseqüentes se mostraram com uma complexidade maior do que se tinha planejado, somado ao pouco tempo para a realização de tais.

Na atual fase de desenvolvimento do sistema, o protótipo realiza as etapas de filtragem e seleção, extração de segmentos de reta, estas duas primeiras fases realizadas por Koser (2003). A fase de Extração de Segmentos de Reta foi melhorada nesta versão na parte que trata da junção de segmentos de retas curtos em segmentos de reta longos, e a fase de Reconhecimento e Interpretação. Estas fases inicias são as fases mais críticas do trabalho por haver pouca literatura sobre os conceitos para a realização delas. Porém as pesquisas para a realização das demais etapas continuam em desenvolvimento.

Em relação aos futuros problemas que se podem encontrar para a realização deste trabalho de pesquisa, encontra-se a incerteza quando se entrada de vídeos de futebol para o sistema é a melhor escolha, pois vídeos de futebol geralmente possuem uma variedade de ruídos na imagem muito grande, devido à iluminação do campo possuir sombras, coloração de gramado entre outros fatores que variam muito de uma imagem para outra. Outro fator bastante relevante é que a quantidade de cenas em uma partida de futebol, que de uma visão da grande área como o sistema exige, são bastante escassas ou são cenas de curta duração.

Quanto ao desempenho computacional em relação à versão anterior do protótipo, praticamente não houve alteração. Este problema não se considera mais tão grave, porque há um constante avanço em relação à velocidade do *hardware*. Foi também visto, mas não implementado, que a Sun (2005) disponibiliza um conjunto de bibliotecas que fazem o processamento de filtragem de imagem, as mesmas filtragens que são usadas neste trabalho,

usando recursos de *hardware* da máquina, como o próprio processamento da placa de vídeo da máquina, o que em teoria melhoria o desempenho do protótipo.

Os números do desempenho do protótipo estão representados na tabela 1.

Tabela 1 -Desempenho

Processador	**Tempo para processar um frame**
Processador 1.8, 256 Ram	0,3 seg
Processador 650, 384 Ram	0,9 seg

A linguagem Java, se mostrou uma boa escolha para a implementação do protótipo pela sua constante evolução, o que permite uma grande variedade de opções para a resolução de um problema.

O protótipo mostrou que a sua grande vantagem é implementar conceitos que podem ser utilizados por outros sistemas, como o próprio sistema do juiz virtual, implementado por Cristofolini (2004), em que se exigia uma interação do usuário para a localização de pontos na imagem para a calibração de câmera, que usando os conceitos deste trabalho pode ser abstraído.

4.1 EXTENSÕES

Como sugestão natural para extensões seria concluir o processo de publicidade virtual como um todo.

Outra sugestão que se considera bastante viável, é aplicar estes conceitos para a realização de publicidade virtual em vídeos de jogos de vôlei de quadra, por exemplo, por fornecer uma qualidade de vídeo bastante regular. Pelo fato de ser filmado em ambientes fechados, existe um certo padrão de iluminação, que facilitaria a interpretação dos segmentos

de reta, as dimensões da quadra não possuem variação, e possuem um modelo real mais simples de ser validado do que o do campo de futebol; as imagens geralmente visualizam a quadra como um todo, que do ponto de vista comercial, dá bastante tempo para vincular publicidade na imagem.

A sugestão anterior não impede que se tente realizar este processo de publicidade virtual para outros esportes como o tênis, futebol americano, entre outros.

O que talvez, no futuro, se faça, é um sistema em que um usuário informe um vídeo de entrada como se fosse um vídeo padrão, o sistema tentaria encontrar um padrão nesta imagem de entrada, com este padrão ele tentaria achar um modelo matemático para imagem inicial, o usuário forneceria um outro vídeo, este agora, onde ele deseja que ocorra a publicidade virtual e com base no modelo realizado na etapa anterior, realizar a publicidade no segundo vídeo de entrada.

REFERÊNCIAS BIBLIOGRÁFICAS

CRISTOFOLINI, Diogo. **Protótipo de um ambiente virtual tridimensional para utilização no cálculo de impedimento de jogadores de futebol.** 2004. 52 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Departamento de Ciências Naturais e Exatas, Universidade Regional de Blumenau, Blumenau.

ECLIPSE. **Eclipse**. [S.l.], [2005?]. Disponível em: <http://www.eclipse.org/>. Acesso em: 26 mai. 2005.

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

GOMES, Paulo César Rodacki. **Desmistificando a publicidade virtua**l. Rio de Janeiro, 1999. 24 f. Guia informativo sobre a tecnologia de publicidade virtual. Publicação interna da Rede Globo de Televisão, Divisão de Engenharia de Multimídia. Central Globo de Engenharia.

KOSER, Everton Elvio. **Reconhecimento automático de linhas de campos de futebol em arquivos de vídeo para publicidade virtual.** 2003. 58 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Departamento de Ciências Naturais e Exatas, Universidade Regional de Blumenau, Blumenau.

QUEIROZ, Gilberto Ribeiro de. **Algoritmos geométricos para banco de dados geográficos da teoria a prática na terralib.** 2003. 147 f. Dissertação (Mestrado em Computação) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos.

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.

SUN Microsystems. [S.l.], [2005?] **The Source for Java Technology**. Disponível em: <http://java.sun.com/>. Acesso em: 3 maio. 2005.