UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSOS DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

CALIBRAÇÃO DE CÂMERAS PARA UTILIZAÇÃO NO CÁLCULO DE IMPEDIMENTOS DE JOGADORES DE FUTEBOL A PARTIR DE IMAGENS

MAYKO STAROSKY

BLUMENAU
2003

2003/2-29

MAYKO STAROSKY

CALIBRAÇÃO DE CÂMERAS PARA UTILIZAÇÃO NO CÁLCULO DE IMPEDIMENTOS DE JOGADORES DE FUTEBOL A PARTIR DE IMAGENS

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 Ciência da Computação — Bacharelado.

Prof. Paulo César Rodacki Gomes

BLUMENAU
2003

2003/2-29

CALIBRAÇÃO DE CÂMERAS PARA UTILIZAÇÃO NO CÁLCULO DE IMPEDIMENTOS DE JOGADORES DE FUTEBOL A PARTIR DE IMAGENS

Por

MAYKO STAROSKY

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. Jomi Fred Hubner, Dr.
Membro:	Prof. Mauro Mattos, Dr.

Blumenau, 09 de dezembro de 2003

Dedico este trabalho à minha esposa pela compreensão, apoio e incentivo, à minha família que sempre acreditou em mim, aos amigos e ao professor orientador que me ajudaram na realização deste.

Há homens que lutam um dia e são bons...
Há outros que lutam um ano e são melhores...
Há os que lutam muitos anos e são muitos bons. Entretanto, há homens que lutam por toda a vida. Estes são imprescindíveis.

Bertold Brecht

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça, que me capacita e fortifica para os desafios.

À minha esposa, pelo amor, compreensão, carinho e apoio em todos os momentos.

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

Aos meus amigos, pelas cobranças e força.

Ao meu orientador, por ter acreditado neste trabalho.

RESUMO

Este trabalho apresenta um método para realizar o cálculo de calibração de câmeras para utilização no cálculo de impedimentos de jogadores de futebol e cálculo de distâncias entre dois pontos no campo de futebol para sistemas do tipo “tira-teima”. O objetivo de tais sistemas é a determinação da posição de impedimento de um jogador de futebol com relação ao seu adversário em imagens estáticas de lances de jogos de futebol. O método proposto utiliza como entrada de dados arquivos de imagem *raster* e um modelo de campo de futebol, onde são informados pontos de referência do campo, gerando um sistema de coordenadas tridimensional para a cena. Após isto o usuário indica a posição dos jogadores de futebol em questão, ou seleciona dois pontos quaisquer do campo , calculando as suas posições no sistema de coordenadas tridimensional, e determinando a sua distância ou impedimento. A validade da proposta é apresentada através da implementação de um protótipo em linguagem C.

Palavras chaves: Visão computacional, calibração de câmeras, computação gráfica.

ABSTRACT

This work presents a method to calculate the cameras calibration to use it in order to calculate the offside position of soccer players and to calculate the distance of two points in the soccer field in virtual referee softwares. The main objective of these softwares is to determinate the offside position of a soccer player in relation to your opponent in raster pictures of soccer games throws during the matches. The offered method uses as entry data pictures files and a model of soccer field, where are informed reference points of the field, generating a tridimentional coordinate system to the scene. After this the user indicates the position of the soccer players in question or selects two other field points, calculating their positions in a tridimentional coordinate system, determining their distance or their offside position. The offer validity is presented through the implementation of a prototype em C language.

Key-Words: Computer vision, camera calibration, computer graphics.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

CBF – Confederação Brasileira de Futebol

IMPA – Instituto de Matemática Pura e Aplicada

PUC – Pontifícia Universidade Católica

TECGraf – Grupo de Tecnologia em Computação Gráfica

UML – Unified Modeling Language

VRML - Virtual Reality Modeling Language

2D – Duas dimensões

3D – Três dimensões

SUMÁRIO

1 INTRODUÇÃO

No meio esportivo, percebe-se cada vez mais a utilização de recursos de computação gráfica. Durante as transmissões televisivas de eventos esportivos, são utilizados os mais variados recursos, desde a inserção de logotipos até placas de patrocinadores. Em partidas de futebol também é aplicado o recurso popularmente conhecido como “tira-teima” em jogadas polêmicas. Tal recurso consiste em um software para análise de distâncias em um campo de futebol, a partir de dados extraídos de imagens do campo, capturadas por câmeras de televisão. Com tecnologias deste tipo, pode-se analisar a posição de impedimento de um jogador, a distância da barreira no momento da cobrança de uma falta, a velocidade da bola em um chute, ou ainda “desenhar” a trajetória da bola em um chute a gol.

A posição de impedimento em uma partida de futebol é definida em CBF (2000), na regra número 11. Ela consiste em definir que um jogador está impedido se o mesmo estiver mais próximo da linha de meta do que o goleiro e o último defensor adversário, no momento de um passe para o mesmo. Este trabalho propõe um método para realizar o cálculo da posição de impedimento em imagens estáticas de cenas de jogos de futebol.

Os elementos de partidas de futebol, assim como todos os objetos do mundo real, são tridimensionais. Quando uma partida de futebol é transmitida pela televisão, as câmeras utilizadas para filmar o jogo “transformam” as imagens de um ambiente de 3 dimensões (mundo real) para um ambiente de 2 dimensões (tela da televisão, monitor de computador, etc.). Posições em uma cena no mundo real podem ser referenciadas por um sistema de coordenadas cartesiano de 3 dimensões. Esta mesma cena é apresentada em um monitor de televisão como uma imagem composta por uma matriz de pontos, onde cada ponto possui uma cor ou intensidade luminosa. Portanto, os pontos da imagem podem ser referenciados por um sistema de coordenadas cartesiano de 2 dimensões.

Neste trabalho, para fazer medições correspondentes a distâncias na cena real (tridimensional), são utilizadas imagens de transmissões televisivas. A partir destas imagens, é necessário um modelo matemático que permita simular o processo de geração da imagem na câmera, para, a partir disso realizar o processo inverso daquele que seria feito normalmente pela câmera. Ou seja, partindo-se do sistema de coordenadas da imagem gerada pretende-se obter coordenadas num sistema tridimensional equivalente àquele imaginado originalmente na

cena real. Este processo é chamado de calibração de câmera e é apresentado na seção 2.4 Algoritmo Para Calibração de Câmeras.

Uma vez definida a calibração da câmera, é possível calcular a posição dos jogadores no mundo real a partir das coordenadas dos jogadores na imagem estática bidimensional. Com a determinação das posições dos jogadores num sistema de coordenadas tridimensional, pode-se calcular a distância entre eles, determinando se um jogador está em posição de impedimento ou não. Pode-se também calcular distâncias entre a bola e o goleiro, entre a bola e o gol ou entre a bola e uma barreira no ato da cobrança de uma falta.

O presente trabalho apresenta a implementação do processo de calibração de câmeras para jogos de futebol e um protótipo do tipo popularmente chamado “tira-teima”, que implementa este processo de calibração e realiza medições de distância no campo de futebol. No protótipo, o usuário interage indicando as coordenadas de referência em uma imagem estática de um jogo de futebol e em um modelo virtual de campo (utilizando o *mouse*), para posteriormente inferir um sistema de coordenadas da câmera. O método abordado neste trabalho consiste em calcular sistemas lineares com os dados de entrada para possibilitar o cálculo da distância entre os jogadores.

1.1 OBJETIVOS DO TRABALHO

Este trabalho tem como objetivo determinar a condição de impedimento de jogadores de futebol em imagens estáticas de jogos de futebol. Como dados de entrada são utilizados pontos provenientes de imagens estáticas extraídas de transmissões de jogos de futebol.

1.2 ESTRUTURA DO TRABALHO

O Capítulo 1 apresenta uma introdução a este trabalho, enfocando as origens para o desenvolvimento do tema, bem como uma breve explanação de todo o processo que deve ser realizado durante a realização do mesmo.

No Capítulo 2 é apresentada a fundamentação teórica deste trabalho, tratando assuntos sobre a posição de impedimento, os tipos de projeção, as câmeras de vídeo e as suas

propriedades, modelos de câmera de vídeo, homografia, assuntos estes que formam a base de entendimento do processo de calibração de câmeras.

No Capítulo 3 são abordados todos os processos envolvidos no desenvolvimento do protótipo, a sua especificação pelo diagrama de casos de uso e diagrama de atividades, a implementação, as técnicas e ferramentas utilizadas, a operacionalidade da implementação, bem como os resultados obtidos.

No Capítulo 4 são apresentadas as conclusões sobre a aplicação do algoritmo proposto, resultados obtidos com a implementação do algoritmo no protótipo. São apresentadas também algumas propostas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentadas considerações sobre os assuntos que serão tratados neste trabalho, tais como: posição de impedimento, tipos de projeção, câmeras de vídeo, propriedades da câmera, posicionamento relativo a cena, centro óptico da lente, modelo de câmera de vídeo, calibração de câmera, algoritmo para calibração de câmera e realização da calibração de câmera.

2.1 POSIÇÃO DE IMPEDIMENTO

A posição de impedimento está definida nas regras de futebol, mas sempre causou grande polêmica, pois o auxiliar do árbitro (popular “bandeirinha”) precisa assinalar o impedimento de um jogador em lances de grande velocidade, e é muito comum o auxiliar (que utiliza como recurso apenas a sua própria visão) assinalar o impedimento erroneamente, ou mesmo deixar um jogador prosseguir a jogada estando em condição irregular. Com os recorrentes erros humanos, torna-se interessante desenvolver uma ferramenta computacional capaz de avaliar o lance, confirmando ou não a marcação do auxiliar do árbitro.

No conjunto de regras que regem o futebol, a regra 11 determina que “a posição de impedimento consiste em um jogador encontrar-se mais perto da linha de meta contrária que a bola e o penúltimo adversário, no momento em que a bola é tocada ou é jogada por um de seus companheiros, tirando vantagem desta posição, interferindo no jogo ou na ação de um adversário” (CBF 2000, p. 42).

A fig. 1 apresenta uma imagem estática de um lance de uma partida de futebol, sem a aplicação de qualquer recurso de um software tira-teima. Já a fig. 2 apresenta a mesma imagem, porém com 2 linhas indicando a posição de um atacante e de um defensor, no mesmo sentido da linha do fundo do campo, gerados pelo tira-teima, apresentando como resultado a distância entre os 2 jogadores, determinando que o atacante está ou não em posição de impedimento em relação ao defensor.

Fonte: Rede Globo de Televisão

FIGURA 1 - Imagem sem o uso do tira-teima

Na fig. 2 verifica-se que a linha indicativa à posição dos jogadores foi “distorcida”, para ser desenhada paralelamente à linha de fundo do campo. No caso das linhas serem inseridas diretamente, de acordo com as coordenadas da imagem, certamente as linhas não representariam as reais posições dos jogadores.

Fonte: Rede Globo de Televisão

FIGURA 2 - Imagem com o uso do tira-teima

Tanto a fig. 1 quanto a fig. 2 apresentam a imagem da partida de futebol de forma “irregular”, isto é, os objetos apresentados na parte inferior da figura aparentam ter um tamanho maior do que os objetos apresentados na parte superior da imagem. Esta “distorção” da imagem ocorre pelo fato da cena estar sendo apresentada em projeção perspectiva, e não

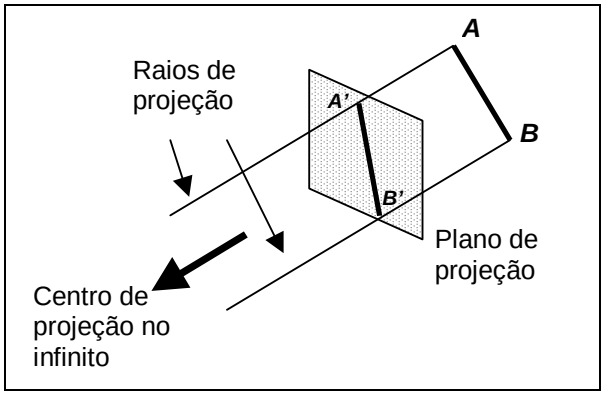
uma projeção paralela (WATT, 1989). Na seção 2.2 - TIPOS DE PROJEÇÃO são apresentados os dois tipos de projeção normalmente utilizados e apresentados na literatura.

2.2 TIPOS DE PROJEÇÃO

Para representar objetos de um mundo 3D em uma tela inerentemente 2D utilizamos uma técnica denominada projeção. Existem dois tipos de projeção:

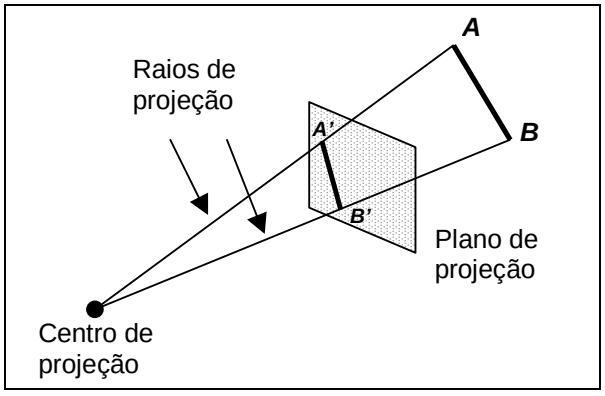
a) projeção paralela: distância do plano de projeção não importa (Fig. 3);

b) projeção panorâmica: reproduz realisticamente efeitos causados pela distância (Fig.4).



Fonte: Wangenheim ([2003?])

FIGURA 3 – Projeção paralela



Fonte: Wangenheim ([2003?])

FIGURA 4 – Projeção perspectiva

No tipo de projeção em perspectiva ocorre uma pirâmide de projeção. Esta projeção é determinada por raios que convergem para o centro de projeção, demonstrado na fig. 4. Na

projeção em perspectiva, os objetos distantes parecem menores, desvanecendo à distância, e os objetos distorcem-se quando vistos de forma oblíqua.

2.3 CÂMERAS DE VÍDEO

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

2.3.1 Propriedades das câmeras de vídeo

Sabe-se que as câmeras possuem propriedades , que uma vez determinadas, transformam uma imagem de um sistema de coordenadas cartesiano de três dimensões para um sistema de coordenadas cartesiano de duas dimensões. Algumas das propriedades das câmeras de vídeo são de grande importância para a aplicação do tira-teima, tais como o posicionamento da mesma em relação à cena e o centro óptico da lente. O modelo de câmera adotado neste trabalho, e descrito na seção 2.3.2, possui 11 parâmetros (5 intrínsecos e 6 extrínsecos).

2.3.1.1 Posicionamento Relativo à Cena

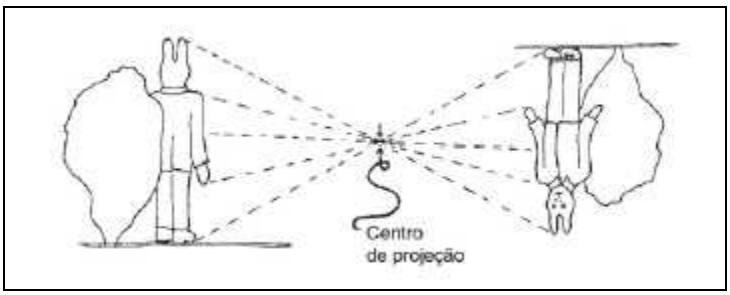
O posicionamento da câmera com relação à cena é a determinação das coordenadas X, Y e Z em um sistema de coordenadas que a câmera ocupa com relação à cena. Não é exatamente uma propriedade de câmera, porém a sua determinação é de fundamental importância para o tira-teima.

2.3.1.2 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 de um sistema do tipo “tira-teima” dependem disso.

2.3.2 Modelo de Câmera de Vídeo

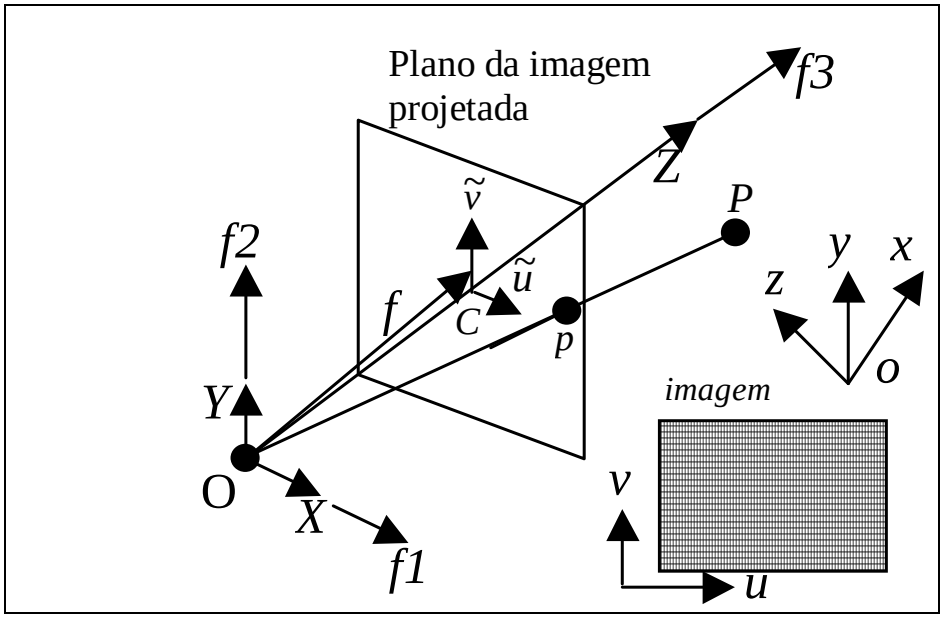
Neste trabalho utiliza-se o modelo de câmera *pinhole*, que é apresentado em Szenberg (2001). Este modelo visa simular o funcionamento de câmeras do tipo *pinhole*, cuja função é capturar as imagens do mundo real que passam por um orifício e são projetadas em um plano. A fig. 5 demonstra como funciona uma câmera *pinhole*.



Fonte: Szenberg (2001, p. 98)

FIGURA 5 - Projeção através de um ponto

Em computação gráfica, esse modelo é representado da forma como demonstra a Fig. 6. O centro de projeção da Fig. 5 corresponde ao ponto $C(\tilde{u},\tilde{v})$ na Fig. 6. O sistema de coordenadas $O(X,Y,Z)$é o sistema de coordenadas do olho humano, que está olhando para o plano da imagem projetada (que pode ser o monitor de uma televisão ou de um computador). O sistema de coordenadas $o(x,y,z)$ é o sistema de coordenadas do mundo real, onde a cena capturada realmente ocorre. Este é o modelo que será utilizado mais adiante neste trabalho para calibrar a câmera.



Fonte: Szenberg (2001, p. 98)

FIGURA 6 - Modelo de câmera Pinhole simplificado

O eixo óptico é definido pelo centro da lente alinhado com o eixo *z*. O plano da imagem, que corresponde ao plano de projeção, $C(\tilde{u}\tilde{v})$ fica perpendicular ao eixo óptico e coincide com o plano *xy*. Neste plano da imagem, é apresentada a imagem da cena desejada em duas dimensões (neste caso, corresponde ao monitor da televisão ou computador). Os pontos neste plano são mapeados por um sistema de coordenadas que possui apenas duas dimensões, *u* e *v*, no sistema de coordenadas cartesiano $C(\tilde{u}\tilde{v})$. O centro óptico *c* está a uma distância focal *f* do plano de imagem, distância esta que interfere diretamente sobre o *zoom* das imagens produzidas.

Ao utilizar o tipo de projeção em perspectiva, o mapeamento de um ponto $P(X,Y,Z)$ do sistema de coordenadas *CXYZ*, para o ponto $p = \tilde{u}\tilde{v}$, é dada pela equação apresentada na fig. 7.

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

Fonte: Szenberg (2001, p. 99)

FIGURA 7 - Mapeamento de pontos em 3D para pontos em 2D

Um ponto em um espaço tridimensional normalmente é apresentando por outro sistema de coordenadas – *oxyz*. Desta forma, para utilizar a equação da fig. 7, para um ponto do sistema de coordenadas *oxyz*, é necessário fazer o mapeamento deste ponto para o sistema de coordenadas da câmera. Isto pode ser feito através do alinhamento dos eixos do sistema de coordenadas *oxyz* e *OXYZ*, compatibilizando as origens. Esta transformação é apresentada na fig. 8.

$$(R \mid T) = \begin{bmatrix} r_1 & t_1 \\ r_2 & t_2 \\ r_3 & t_3 \end{bmatrix} = \begin{bmatrix} r_{1x} & r_{1y} & r_{1z} & t_x \\ r_{2x} & r_{2y} & r_{2z} & t_y \\ r_{3x} & r_{3y} & r_{3z} & t_z \end{bmatrix}$$

Fonte: Szenberg (2001, p. 99)

FIGURA 8 – Compatibilização das origens de sistemas de coordenadas diferentes

Os parâmetros *r1*, *r2* e *r3* definem a direção e a inclinação da câmera, enquanto os parâmetros *t1*, *t2* e *t3* representam a origem do sistema de coordenadas *oxyz*, representada no sistema de coordenadas da câmera.

Feita a compatibilização entre os sistemas cartesianos de coordenadas *oxyz* e *CXYZ*, pode-se dizer que, para mapear um ponto do mundo tridimensional (*x*,*y*,*z*), este é projetado no ponto $(\tilde{u}, \tilde{v})$ através da equação da fig. 9.

$$\begin{bmatrix} \tilde{u}s \\ \tilde{v}s \\ s \end{bmatrix} = \begin{bmatrix} fr_1 & ft_x \\ fr_2 & ft_y \\ r_3 & t_z \end{bmatrix}$$

Fonte: Szenberg (2001, p. 99)

FIGURA 9 – Projeção de ponto do mundo 3D no ponto $(\tilde{u}, \tilde{v})$ 2D

A equação apresentada na fig. 9 também pode ser representada após ser desenvolvida, como é mostrado nas fig. 10 e fig. 11.

$$u - u_0 = f \frac{r_{1x}x + r_{1y}y + r_{1z}z + t_x}{r_{3x}x + r_{3y}y + r_{3z}z + t_x}$$

Fonte: Szenberg (2001, p. 99)

FIGURA 10 – 1ª equação da projeção de ponto do mundo 3D no ponto $(\tilde{u}, \tilde{v})$ 2D

$$v - v_0 = f \frac{r_{2x}x + r_{2y}y + r_{2z}z + t_x}{r_{3x}x + r_{3y}y + r_{3z}z + t_x}$$

Fonte: Szenberg (2001, p. 99)

FIGURA 11 - 2ª equação da projeção de ponto do mundo 3D no ponto $(\tilde{u}, \tilde{v})$ 2D

As equações da fig. 10 e fig. 11 podem ser escritas em forma matricial, de apresentado na fig. 12.

$$\begin{bmatrix} us \\ vs \\ s \end{bmatrix} = Q \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} q_1 & q_{14} \\ q_2 & q_{24} \\ q_3 & q_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} fr_1 + u_0 r_3 & ft_1 + u_0 t_z \\ fr_2 + v_0 r_3 & ft_y + v_0 t_z \\ r_3 & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Fonte: Szenberg (2001, p. 100)

FIGURA 12 – Forma matricial das equações de projeção

Na fig. 13 é apresentado o sistema de equações lineares utilizado para fazer o mapeamento de pontos do mundo real (3D) para o plano da imagem (2D) através da transformação projetiva genérica, evitando assim as restrições impostas pelo modelo de câmera "*pinhole*". Antes de executar o mapeamento descrito, é necessário efetuar a calibração da câmera, que consiste em calcular todos os valores q_{11} a q_{34}, utilizando coordenadas homogêneas.

$$\begin{bmatrix} us \\ vs \\ s \end{bmatrix} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Fonte: Szenberg (2001, p. 100)
FIGURA 13 – Matriz de projeção genérica

Uma vez efetuada a calibração da câmera, é possível mapear pontos do sistema de coordenadas *u*,*v* (modelo de campo) para o sistema *x*,*y*,*z* (mundo real), definindo assim as posições reais de objetos na imagem (neste trabalho, os jogadores) e aplicar desta forma o tira-teima.

Existem casos onde os pontos do mundo tridimensional que serão utilizados para calibração da câmera são todos coplanares, isto é, estão no mesmo plano. Isto significa que é possível efetuar uma simplificação do sistema proposto na fig. 13. Este tipo de calibração de câmeras é chamado de Homografia, e é descrito na próxima seção.

2.3.3 Homografia

Em muitas situações os pontos do mundo tridimensional estão todos no mesmo plano, como por exemplo, pontos no campo de futebol. Se os pontos são todos coplanares, isto significa que os mesmos possuem a coordenada *z* igual a zero. Tendo esta condição, pode-se simplificar a equação apresentada na fig. 13, como exposto na fig. 14.

$$\begin{bmatrix} us \\ vs \\ s \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Fonte: Szenberg (2001, p.101)
FIGURA 14 – Transformação projetiva planar (homografia)

Esta transformação é uma transformação projetiva planar, também conhecida como homografia. O processo para encontrar uma homografia é idêntico ao processo de calibração de câmeras, normalmente utilizado quando a coordenada *z* for maior do que zero. Como definição de calibração de câmera, pode-se dizer que "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" (SZENBERG, 2001).

O processo de encontrar uma homografia consiste em uma maneira de transformar um sistema de coordenadas lógico no próprio sistema de coordenadas da câmera. Este processo é fundamental para a aplicação do tira-teima, pois utiliza-se imagens estáticas capturadas de transmissões televisivas, não possuindo desta forma os reais parâmetros da câmera. Uma boa homografia permite definir a velocidade da bola ou a verificação de impedimentos com grande precisão.

Segundo Szenberg (2001),

> Tomando um quadro de um de vídeo obtido através da televisão, será determinado um modelo de câmera e aplicados cálculos para a determinação da transformada inversa da imagem daquele quadro (parte-se de uma imagem em duas dimensões (2D), calculando a sua transformada inversa para criar a situação relativa em três dimensões (3D), calculando neste ponto as distâncias entre os jogadores de futebol para definir o impedimento ou não, bem como verificar a distância da bola até a meta adversária). Para que a inserção de objetos virtuais em imagens reais possa ser realizada, é necessário conhecer a câmera, obtendo informações como sua posição, orientação e inclinação (parâmetros extrínsecos), além de características de seu sistema óptico (parâmetros intrínsecos). A partir destas informações e das posições projetadas dos objetos na cena de pontos da imagem real, é possível fazer medições. Muitas vezes, quando todos os elementos de interesse estão em um único plano, basta encontrar uma transformação projetiva planar para atender a esses objetivos. Então, o problema de inserir objetos virtuais em imagens reais pode ser colocado como um problema de determinação dos parâmetros da câmera e sobreposição ou combinação de imagens – imagens reais mais imagens virtuais. Uma vez conhecida a câmera, podemos modelar a cena 3D a partir da imagem.

Para que possa ser encontrada a homografia, é necessário estudar todo o processo efetuado pelas câmeras. Pode-se ver na fig. 13 a matriz de projeção, que mapeia os pontos de coordenadas do mundo 3D para coordenadas 2D.

Com a simplificação proposta, é necessário calcular os parâmetros da transformação. A fig. 15 apresenta a matriz H, com os parâmetros da transformação que devem ser calculados.

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Fonte: Szenberg (2001, p.101)
FIGURA 15 – Parâmetros da homografia

2.4 ALGORITMO PARA CALIBRAÇÃO DE CÂMERAS

O processo de calibração de câmeras aplicado neste trabalho, utiliza como entrada uma imagem estática de uma cena de um jogo de futebol e uma imagem estática de um modelo de campo de futebol. Com estas 2 imagens, deve-se informar os pontos de referência (utilizando o *mouse*), clicando-se na imagem do jogo de futebol e em seguida no modelo de campo de futebol, em pontos predefinidos (as interseções das linhas do campo, com exceção dos dois semi-círculos nas grandes áreas). Como foi exposto na seção 2.3.3, para que a calibração seja efetuada corretamente, é necessário que as referências de campo sejam todas coplanares, isto é, todas as referências indicadas devem estar no campo de futebol, excluídos os cantos superiores das traves. Com todas as referências marcadas, é feito o cálculo da homografia apresentada na fig. 14, sendo esta a principal dificuldade encontrada neste trabalho. Para realizar a operação de homografia, deve-se resolver o sistema de equações lineares cujo resultado consiste no conjunto de parâmetros da câmera.

Para explicar em detalhes o processo realizado, 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.

2.4.1 Realização da calibração de câmeras

Esta seção demonstra como utilizar os dados encontrados na seção anterior para calibrar uma câmera de *Pinhole*, vista na seção 2.3.2 Modelo de Câmera de Vídeo e seção 2.3.3 Homografia.

Para poder resolver as equações encontradas, deve-se selecionar 4 (quatro) pares de pontos de referência, que tenham as posições determinadas tanto no sistema $C(\tilde{u},\tilde{v})$, quanto no Sistema do Universo U(*x*,*y*,*z*). Esses pontos são locais predeterminados no campo de futebol onde, num primeiro momento, suas coordenadas *x*, *y* e *z* são conhecidas, como a marca do pênalti e interseções das linhas do campo. É importante ressaltar que neste trabalho utiliza-se o número mínimo de pontos para calcular a homografia, ocasionando menor precisão nas medidas obtidas como resultado. Quanto maior o número de pontos de referência utilizados para fazer a calibração de câmeras, mais precisa será a homografia calculada.

Após a seleção dos 4 pares de pontos, tem-se um sistema linear de equações com 12 variáveis e 12 equações, resultantes de 3 restrições para cada par de pontos selecionados.

$$\begin{bmatrix}
x_1 & y_1 & w_1 & 0 & 0 & 0 & 0 & 0 & 0 & -u_1 & 0 & 0 \\
0 & 0 & 0 & x_1 & y_1 & w_1 & 0 & 0 & 0 & -v_1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & x_1 & y_1 & w_1 & -t_1 & 0 & 0 \\
x_2 & y_2 & w_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -u_2 & 0 \\
0 & 0 & 0 & x_2 & y_2 & w_2 & 0 & 0 & 0 & 0 & -v_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & x_2 & y_2 & w_2 & 0 & -t_2 & 0 \\
x_3 & y_3 & w_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -u_3 \\
0 & 0 & 0 & x_3 & y_3 & w_3 & 0 & 0 & 0 & 0 & 0 & -v_3 \\
0 & 0 & 0 & 0 & 0 & 0 & x_3 & y_3 & w_3 & 0 & 0 & -t_3 \\
x_4 & y_4 & w_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & x_4 & y_4 & w_4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & x_4 & y_4 & w_4 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \\ s_1 \\ s_2 \\ s_3
\end{bmatrix}
=
\begin{bmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ u_4 \\ v_4 \\ 1
\end{bmatrix}$$

Fonte: Szenberg (2001, p. 102)

FIGURA 16 - Sistema de equações para determinar os parâmetros da câmera

2.5 MÉTODO DE GAUSS-JORDAN

A partir do momento em que o usuário selecionou 4 pares de pontos de referências (4 pontos na imagem estática obtida de uma partida de futebol e os respectivos 4 pontos no modelo do campo), está montado o sistema de equações lineares apresentado na fig. 16. Para fins de apresentação, simplifica-se o sistema de equações lineares apresentado na fig. 16 como sendo $AC = B$. Um dos objetivos do protótipo é encontrar a solução do sistema, que consiste em determinar os valores para os elementos do vetor de incógnitas *C*. Para resolver o sistema de equações lineares apresentado, é utilizado o método de Gauss-Jordan, também conhecido como eliminação gaussiana.

Segundo Lawson (1996), a eliminação gaussiana consiste em "simplificar um conjunto de equações por eliminação de uma variável de cada vez de equações sucessivas". Isto significa que o sistema de equações é manipulado integralmente, calculando assim os valores do vetor *C*, elemento a elemento, resultando nos 9 parâmetros da homografia (q_{11} a q_{33}).

O método de Gauss-Jordan é um método de resolução de sistemas de equações lineares de simples compreensão e muito eficiente.

$$A.B = \left[\begin{array}{ccccc|c} a_{11} & a_{12} & \ldots & a_{1m} & | & b_1 \\ a_{21} & a_{22} & \ldots & a_{2m} & | & b_2 \\ -- & -- & -- & -- & -- & -- \\ a_{m1} & a_{m2} & \ldots & a_{mn} & | & b_m \end{array}\right]$$

FIGURA 17 - Matriz ampliada do sistema de equações

Para resolver sistemas de equações lineares pelo método de Gauss-Jordan, deve-se seguir a seqüência de passos abaixo, segundo Press (1988):

a) ao lado da matriz dos coeficientes das variáveis (*A*), coloca-se a matriz-coluna dos termos independentes (*B*). A matriz resultante é denominada matriz ampliada do sistema, onde cada linha da matriz representa de forma abreviada a equação correspondente no sistema. A fig. 17 apresenta a matriz ampliada do sistema de equações lineares.

b) por meio das operações elementares (permutação de linhas, multiplicação de uma linha inteira por um escalar, soma de um múltiplo de uma linha a uma outra linha) sobre

as linhas da matriz ampliada, transforma-se a matriz ampliada em uma matriz em forma escalonada.

A matriz em forma escalonada calculada neste trabalho, pode ser classificada como uma matriz triangular superior. Segundo Lawson (1997), “se todos os elementos abaixo da diagonal (elementos cujos índices da linha é igual ao índice da coluna) são nulos, a matriz se diz triangular superior”. A fig. 18 apresenta uma matriz triangular superior.

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \ldots & a_{1m} & b_1 \\ 0 & a_{22} & \ldots & a_{2m} & b_2 \\ -- & -- & -- & -- & -- \\ 0 & 0 & \ldots & a_{mn} & b_m \end{array}\right]$$

FIGURA 18 - Matriz triangular superior

Estando a matriz na forma triangular superior, a estratégia utilizada é resolver as equações de baixo para cima, resolvendo uma variável de cada vez. Este processo é denominado de “substituição para trás” (LAWSON, 1997). Desta forma, a equação a_{mn} (que é a última linha do sistema) está calculada e é igual a b_m. Com o valor de a_{mn} determinado, faz-se a substituição deste elemento na equação da linha imediatamente acima, possibilitando calcular o valor de $a_{m-1\ n-1} = b_m - a_{mn}$. Este processo é realizado sucessivamente, até serem calculados os valores de todas as equações, que correspondem aos valores do vetor *C*.

2.6 IMAGENS RASTER

A imagem gráfica com formato *raster* é a mais comum tecnologia de representação de imagens em uso na atualidade, pois torna possível simular os efeitos de cor, luz e sombra nos objetos realísticos e permite manipular o menor detalhe da figura. As figuras normalmente especificadas por este meio são imagens captadas através de *scanners* ou câmeras, para serem modificadas através da aplicação de técnicas de manipulação de suas características originais. A esta manipulação chama-se de processamento de imagens (CHANG, 1989).

Para utilização pelo protótipo, são aceitos os mais variados formatos de arquivos de imagens raster, entre elas BMP, PCX, JPG, TIF, TGA.

2.7 SISTEMAS ATUALMENTE EXISTENTES

Atualmente existem vários sistemas que extraem informações de imagens, especialmente de imagens esportivas, entre eles alguns sistemas do tipo “tira-teima”. Uma breve descrição sobre o funcionamento dos sistemas comerciais mais utilizados é apresentada na tabela 1.

Tabela 1 – Descrição dos sistemas comerciais utilizados para tira-teima

Estrutura	**Elemento**
VirtuaLive	Permite criar animações em formato VRML a partir de uma seqüência de vídeo curta de uma partida de futebol. Os jogadores são representados por polígonos
Digital Replay	Realiza o acompanhamento de atletas em uma seqüência de imagens, podendo, entre outras coisas, calcular medidas em um campo de futebol, determinar a velocidade dos jogadores e da bola e mostrar a linha de impedimento em transmissões de partidas de futebol.
VirtualReplay	Produz uma reconstrução 3D de uma imagem de um jogo de futebol. Uma interação manual é necessária para criar os jogadores, que são representados por polígonos, utilizando tecnologia de *bill boar*d. Como o sistema opera com uma imagem tomada de uma única câmera, não é possível calcular a posição 3D da bola e dos jogadores quando não estão na altura do gramado.

Fonte: Szenberg (2001, p. 22 ss.).

Vale citar o sistema desenvolvido e utilizado pela Rede Globo de Televisão, onde, com base em transmissões de jogos de futebol, são informadas referências do campo e jogadores, gerando assim um modelo virtual da cena. Desta forma, baseado no modelo virtual, é calculada a distância entre 2 jogadores no momento de um lançamento, a distância da bola até a meta ou mesmo a distância da bola até a barreira em uma falta, além de calcular a velocidade da bola após um chute a gol ou cobrança de falta. Outro exemplo é o Juiz Virtual (IMPA, [2003?]), aplicativo que utiliza técnicas de modelagem baseada em imagens, onde são informados pontos de referência na imagem da partida de futebol e em um modelo virtual a parte, permitindo desta forma calibrar a câmera e depois calcular as distâncias de impedimento.

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 PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Para realizar a determinação do impedimento deve-se passar para o sistema um arquivo de imagem estática contendo uma cena de um jogo de futebol e alguns pontos de referência do campo. Após isto, é calculada a matriz de transformação da imagem e informando a posição do jogador de ataque e do jogador de defesa, determina-se a distância entre os dois jogadores, calculando assim a posição de impedimento.

O sistema deve retornar a distância entre 2 jogadores de futebol em um determinado lance, conforme as regras de futebol. Pode-se também determinar a distância da bola até algum outro ponto da cena, como por exemplo a barreira, o goleiro, etc.

3.2 ESPECIFICAÇÃO

Para fazer a especificação do protótipo, foi utilizada a linguagem de modelagem UML, descrita por Quatrani (2001). Como diagramas da especificação serão apresentados o diagrama de casos de uso e o diagrama de atividades do protótipo. Para gerar a modelagem do protótipo, foi utilizada a ferramenta Rational Rose.

3.2.1 DIAGRAMA DE CASOS DE USO

Para fazer a especificação do protótipo, foi feito o Diagrama de Casos de Uso, apresentado na fig. 19.

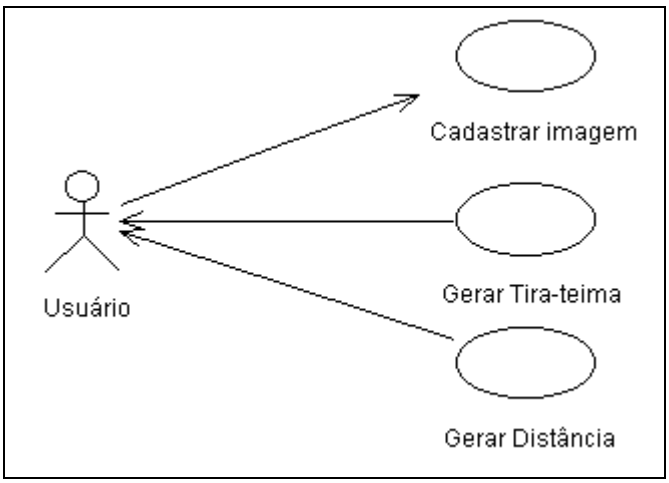


FIGURA 19 - Diagrama de Casos de Uso

Para obter os resultados desejados, o Usuário deve cadastrar uma imagem contendo uma cena de uma partida de futebol e o protótipo deve gerar, como retorno para o usuário, a posição ou não de impedimento dos jogadores de futebol na imagem com a respectiva distância calculada entre os jogadores, ou ainda informar a distância entre dois pontos no campo (como da bola até a barreira no momento da cobrança de uma falta, por exemplo).

3.2.2 DIAGRAMA DE ATIVIDADES

A fig. 20 apresenta o diagrama de atividades, representando o fluxo básico das atividades do protótipo, que é o cálculo do tira-teima.

Uma vez iniciado o protótipo, o usuário deve abrir uma imagem estática de um lance de uma partida de futebol. Estando esta imagem carregada, o mesmo deve clicar no botão que indica que serão marcadas as referências do campo. Neste momento o usuário deve indicar quatro (4) referências na imagem da partida de futebol e sua respectiva referência no modelo de campo apresentado pelo protótipo.

Com todas as referências de campo assinaladas, é calculada a transformada inversa, calculando a câmera que gerou a imagem da partida de futebol.

Neste ponto o usuário deve optar por medir o impedimento, ou calcular a distância entre 2 pontos que ele indicar. Clicando no botão de referência do atacante, o usuário deve clicar na imagem da partida de futebol para selecionar a referência do mesmo na imagem. Automaticamente serão calculadas as coordenadas deste jogador no sistema de coordenadas em três dimensões (3D). Em seguida, o usuário deve selecionar a referência do defensor na imagem da partida de futebol e também será calculada a sua posição no mundo real.

Calculadas as posições no mundo real dos jogadores, é calculada a diferença da distância entre os jogadores, caracterizando o impedimento ou não.

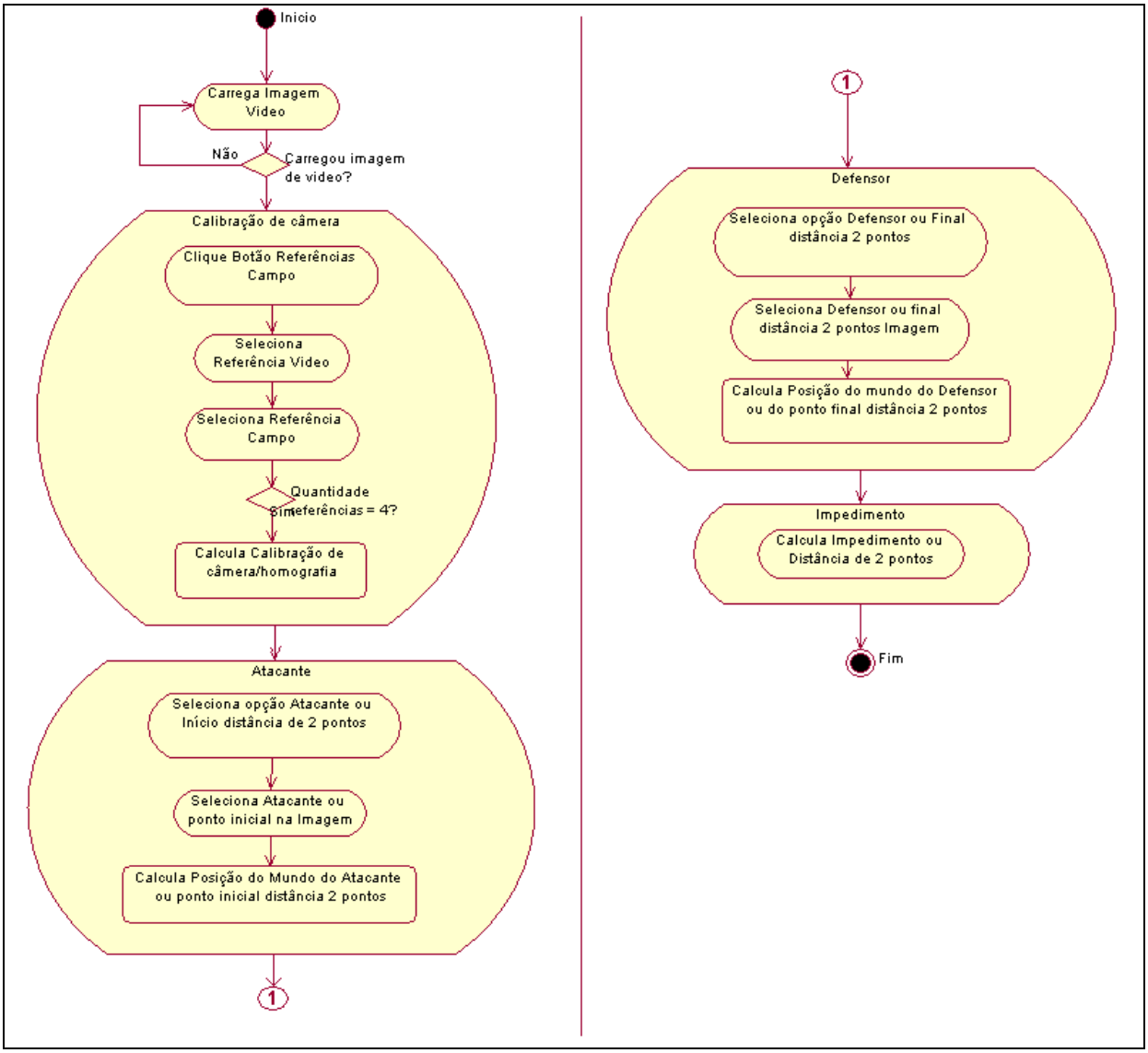


FIGURA 20 – Diagrama de atividades

3.3 IMPLEMENTAÇÃO

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

3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

Para a implementação do protótipo foram utilizados o ambiente de desenvolvimento Visual C++ 6.0 da Microsoft, em conjunto com as bibliotecas de geração de interface IUP (TECGRAF, 2003), de apresentação gráfica IM (TECGRAF, 2003?) e CD (TECGRAF,

2003?), além de uma rotina onde é aplicado o método de Gauss-Jordan para a solução de sistemas de equações lineares. O Visual C++ foi utilizado tendo em vista o fato de ser uma ferramenta de domínio do autor, e as demais bibliotecas porque são ferramentas específicas para o tratamento de imagens e cálculo de sistemas de equações lineares, e englobam tratamento para utilizar vários formatos de imagens existentes no mercado, além de recursos para a realização dos cálculos necessários.

Informações sobre o Visual C++ são encontradas em Kruglinski (1998). Informações sobre as bibliotecas IUP, CD, IM são encontradas em TECGraf (2003?).

O protótipo possui em sua interface uma área para a apresentação da imagem a analisar o possível impedimento e uma área com um modelo de campo de futebol. Este modelo de campo de futebol possui uma série de pontos predefinidos para a marcação das referências (os pontos de referências são todas as interseções das linhas delimitadoras do campo, com exceção do semicírculo das grandes áreas). No quadro 1 encontra-se um trecho de código C que contém a função principal do protótipo.

```
/* Função principal do protótipo */
int main(void)
{
	char *error=NULL;
	Ihandle *jnprincipal = NULL;
	IupOpen();
.....................................................................
/* Função que lê o arquivo com a definição da interface do protótipo */
	if (error = IupLoad ("tirateima.led"))
	{
		IupMessage ("LED Error", error);
		return 1;
	}
	.....................................................................
/* Cria a janela do protótipo */
	jnprincipal = IupGetHandle("jnprincipal");
	IupShow(jnprincipal);
	IupSetAttribute(jnprincipal,IUP_CLIPCHILDREN,IUP_NO);
.....................................................................
/* Carrega a imagem do modelo do campo de futebol */
if (carrega_campo(1,0) == 1)
{
	IupClose();
		return 0;
}
/* Rotina de loop em que o protótipo fica após ser criado */
IupMainLoop();
/* Função para finalizar o protótipo e liberar a memória */
IupClose();
return 0;
}
```

QUADRO 1 - Código de inicialização do protótipo

O usuário deve, primeiramente, abrir uma imagem de uma partida de futebol para depois informar as referências do campo, clicando no botão que indica que serão marcadas estas referências. Indicado o referido botão, deve-se clicar na imagem que ele cadastrou e no ponto correspondente no modelo de campo de futebol. No quadro 2 encontra-se um trecho de código C que recebe os pontos selecionados na imagem que o usuário cadastrou.

```
/* Função executada quando é clicado na área de vídeo */
int click_video_cb(Ihandle* iup_Canvas_Video, int button, int estado, int u, int v)
{
.....................................................................
/* Verifica se pode ser marcada a referência no vídeo – somente após ter clicado no
botão de referências de vídeo */
if ((Status_Video == 0) && (Status_Campo == 0) && (Modo_Referencias_Campo == 1))
{
      /* Verifica se o botão esquerdo do mouse foi pressionado */
      if (button == IUP_BUTTON1)
      {
         Pos_u_Video = u - (gb/2);           /* gb = Global Base */
         Pos_v_Video = (300 - v) - (ga/2);   /* ga = Global Altura */
         Status_Video = 1;      /* Indica que referência de vídeo foi marcada */
.....................................................................
```

QUADRO 2 - Código da rotina que trata o clique na imagem estática do lance de futebol

Após o usuário indicar a referência do campo, com um clique do mouse, é necessário que ele indique a referência no modelo de campo de futebol. Como já exposto, as referências no modelo do campo estão localizadas nas interseções das linhas do campo.

Para acelerar a localização da referência selecionada, o modelo do campo de futebol sofreu uma divisão em sete (7) regiões. Desta maneira, evita-se que, ao selecionar a referência do *corner* inferior direto, seja feito o teste em todas as outras referências do campo primeiro. Assim, quando o usuário clica no modelo de campo, é verificado em qual a região foi efetuado o clique do mouse, para que somente após identificada a região, seja localizada a referência. A fig. 21 apresenta a forma de divisão em regiões aplicado no modelo de campo de futebol neste protótipo.

FIGURA 21 – Exemplo de divisão do campo em regiões

No quadro 3 encontra-se um trecho de código C que recebe os pontos selecionados na imagem que o usuário cadastrou, faz a checagem em qual região foi clicado e depois localiza a referência da região que foi selecionada.

```
int click_campo_cb(Ihandle* iup_Canvas_Campo, int button, int estado, int u,int v)
{
.............................................................
v = 118 - v;    /* Inversão de direção do eixo v do sistema de coordenadas */
        /* Verifica a região em que o ponto que foi clicado no modelo de campo */
  if (((u >= 0) && (u <= 115)) && ((v >= 88) && (v <= 118)))
        {
          /* Verifica em que área do campo o ponto que foi selecionado */
/* Os valores 69 e 111 são os pontos exatos da referência, mas é deixada uma margem
de 5 pontos em volta do ponto exato para aceitar o clique */
          if (((u >= 69-5) && (u <= 69+5)) && ((v >= 111-5) && (v <= 111+5)))
          {
             /* Adiciona os valores na matriz de projeção */
             Adiciona_Item_Matriz ( 0.0, 75.0, 0.0, Qde_Pontos_Selecionados);
             Qde_Pontos_Selecionados++;  /* Incrementa o número de pontos selec.*/
             Altera_Mensagem (Qde_Pontos_Selecionados);
             Status_Campo = 1;   /* Indica que já selecionou referência */
          }
          else
    {  /* Os valores 58 e 93 são os pontos exatos da referência, mas é deixada uma
margem de 5 pontos em volta do ponto exato para aceitar o clique */
       if (((u >= 58-5) && (u <= 58+5)) && ((v >= 93-5) && (v <= 93+5)))
             {
               Adiciona_Item_Matriz ( 0.0, 57.66, 0.0, Qde_Pontos_Selecionados);
               Qde_Pontos_Selecionados++;
               Altera_Mensagem (Qde_Pontos_Selecionados);
               Status_Campo = 1;
             }
             else
             { /* Os valores 98 e 93 são os pontos exatos da referência, mas é
deixada uma margem de 5 pontos em volta do ponto exato para aceitar o clique */

                 if (((u >= 98-5) && (u <= 98+5)) && ((v >= 93-5) && (v <= 93+5)))
                 {
                    Adiciona_Item_Matriz(16.50,57.66,0.0, Qde_Pontos_Selecionados);
                    Qde_Pontos_Selecionados++;
             Altera_Mensagem (Qde_Pontos_Selecionados);
                    Status_Campo = 1;
                 }
             }
         }
```

QUADRO 3 - Código da rotina que trata o clique no modelo de campo de futebol

Após efetuar-se a seleção dos 4 pontos de referência do campo, são calculadas as raízes do sistema de equações lineares apresentado na fig.10. Uma vez calculado o sistema de equações, está determinada a homografia.

No quadro 4 encontra-se um trecho de código C que faz o cálculo dos parâmetros da câmera (homografia).

```
void Gauss_Jordan(matrix15 a, double *c, double *x, int n)
{
double pitemp,pidum,pibig,am,susum;
int i,j,ell,i1,elx,pij,i2,sul,ix,sui,suj,jx;
/* FORWARD ELIMINATION */
       ell=n-1;
       for (i1=1;i1<=ell;i1++)
       {
              elx=i1+1;
              pij=i1;
              pibig=fabs(a[i1][i1]);
              for (i=elx;i<=n;i++)
              {
                     am=fabs(a[i][i1]);
                     if (am>pibig)
                     {
                            pibig=am;
                            pij=i;
                     }
              }
              if (pij != i1)
              {
                     for (j=i1;j<=n;j++)
                     {
                            pidum=a[pij][j];
                            a[pij][j]=a[i1][j];
                            a[i1][j]=pidum;
                     }
                     pitemp=c[pij];
                     c[pij]=c[i1];
                     c[i1]=pitemp;
              }
              for (i2=elx;i2<=n;i2++)
              {
                     for (j=elx;j<=n;j++)
                            a[i2][j]=a[i2][j]-a[i2][i1]/a[i1][i1]*a[i1][j];
                     c[i2]=c[i2]-a[i2][i1]/a[i1][i1]*c[i1];
              }
       } /* NEXT i1 */
       /* BACK SUBSTITUTE */
       sul=n-1;
       x[n]=c[n]/a[n][n];
       for (ix=1;ix<=sul;ix++)
       {
              susum=0;
              sui=n-ix;
              suj=sui+1;
              for (jx=suj;jx<=n;jx++)
                     susum=susum+a[sui][jx]*x[jx];
              x[sui]=(c[sui]-susum)/a[sui][sui];
       }
}
```

Fonte: Press (1988).

QUADRO 4 - Código da rotina que calcula os parâmetros da homografia

Após a determinação dos parâmetros de câmera (homografia), deve-se indicar na imagem da partida de futebol, onde está o atacante (se o usuário deseja calcular o

impedimento) ou a posição da bola (para calcular a distância da bola até outro ponto no campo).

Se o usuário deseja verificar a posição de impedimento de um jogador, deve então clicar no botão que indica que será selecionada a referência do atacante, para em seguida ser indicado com o uso do mouse, na imagem da partida de futebol, a localização do atacante. Após a indicação do atacante, são calculadas as coordenadas do jogador em pontos do mundo real (3D), resolvendo o sistema de equações lineares, exposto na fig. 16. Este sistema de equações é resolvido aplicando o método de Gauss-Jordan, da mesma forma que o sistema de equações para determinar os parâmetros da câmera.

Da mesma forma, se o usuário deseja calcular alguma distância a partir da bola, deve-se clicar no botão que indica que será selecionada a posição da bola, para depois indicá-la na imagem da partida de futebol. No quadro 5 encontra-se um trecho de código C que faz o cálculo da posição no mundo do atacante ou da bola.

```
      /* Cria a matriz com os parâmetros da câmera */
         Cam[0][0] = C[1];
         Cam[0][1] = C[2];
         Cam[0][2] = C[3];
         Cam[1][0] = C[4];
         Cam[1][1] = C[5];
         Cam[1][2] = C[6];
         Cam[2][0] = C[7];
         Cam[2][1] = C[8];
         Cam[2][2] = C[9];
      /* Cria matriz auxiliar com os parâmetros da câmera */
         for (i=0; i<3; i++)
         {
           for(j=0 ;j<3; j++)
             Cam_aux[i+1][j+1]= Cam[i][j];
         }
      /* Cria o vetor com as coordenadas 2D do ponto clicado no vídeo */
         Ti[0] = u - (gb/2);
         Ti[1] = (300 - v) - (ga/2);
         Ti[2] = 1.0;
      /* Cria o vetor auxiliar com as coordenadas 2D do ponto clicado no vídeo */
         for (i=0; i<3; i++)
                Ti_aux[i+1]= Ti[i];
      /* Aplica o método de Gauss-Jordan para resolver o sistema de equações
         e determinar as coordenadas 3D correspondentes ao ponto 2D */
         Gauss_Jordan(Cam_aux,Ti_aux,Pw,3);
/* Determina a posição real nas coordenadas do mundo do atacante/ponto inicial */
         Posicao_atacante_x = Pw[1]/Pw[3];
         Posicao_atacante_y = Pw[2]/Pw[3];
         if (Modo_Referencias_Atacante == 1)
         {
            Modo_Referencias_Atacante = 2;
        Desativa_botoes(4);
         }
         else
         {
           if (Modo_Referencias_Bola == 1)
           {
             Modo_Referencias_Bola = 2;
             Desativa_botoes(8);
           }
      }
```

QUADRO 5 - Código de cálculo das coordenadas do mundo do atacante e bola

Após a determinação das coordenadas do mundo do atacante ou da bola, é feito processo semelhante para calcular a posição do defensor ou do ponto destino da medida de distância a partir da bola. No quadro 6, encontra-se um trecho de código C que faz o cálculo da posição no mundo do atacante.

```
       /* Recria matriz auxiliar com os parâmetros da câmera */
       for (i=0; i<3; i++)
       {
          for(j=0 ;j<3; j++)
              Cam_aux[i+1][j+1]= Cam[i][j];
       }
       /* Adiciona ao vetor as coordenadas 2D do novo ponto selecionado */
       Ti[0] = u - (gb/2);
       Ti[1] = (300 - v) - (ga/2);
      Ti[2] = 1.0;
       /* Adiciona ao vetor auxiliar as coordenadas 2D do novo ponto selecionado */
       for (i=0; i<3; i++)
       {
          Ti_aux[i+1]= Ti[i];
       }

       /* Aplica o método de Gauss-Jordan para resolver o sistema de equações
          e determinar as coordenadas 3D correspondentes ao ponto 2D */

       Gauss_Jordan(Cam_aux,Ti_aux,Pw,3);

/* Determina a posição real nas coordenadas do mundo do defensor/ponto final */
       Posicao_defensor_x = Pw[1]/Pw[3];
       Posicao_defensor_y = Pw[2]/Pw[3];

       if (Modo_Referencias_Defensor == 1)
       {
              Modo_Referencias_Defensor = 2;
              Desativa_botoes(5);
       }
       else
       {
              if (Modo_Referencias_Barreira == 1)
              {
                     Modo_Referencias_Defensor = 2;
                     Desativa_botoes(9);
              }
       }
.....................................................................
```

QUADRO 6 - Código de cálculo das coordenadas do mundo do defensor/barreira

Determinadas as coordenadas dos jogadores de ataque e defesa (ou da bola e do ponto de destino) no sistema de coordenadas do mundo, subtrai-se a variável *posicao_atacante*, que recebeu o valor da coordenada *Pw[1]* do atacante (correspondente à coordenada *x* do atacante nas coordenadas do mundo 3D) pela variável *posicao_defensor*, que recebeu o valor da coordenada *Pw[1]* do defensor (correspondente à coordenada *u* do defensor no sistema de coordenadas do mundo 3D) e determina-se a distância entre eles. No quadro 7 encontra-se um trecho de código C que calcula a posição de impedimento do atacante com relação ao defensor.

```
/* Rotina para calcular a posição de impedimento ou não de jogadores */
void Calcula_Impedimento (double Atacante, double Defensor)
{
	distancia = 0.00;

	/* verifica em qual lado do campo está sendo feita a verificação */
	if ((Atacante > 55) || (Defensor > 55))
	{
		/* lado direito do campo */
		distancia = Atacante - Defensor;
		/* conversão de sinal, para deixar o sinal sempre positivo */
		if (distancia < 0)
			distancia = distancia * (-1);
		/* Verifica se está em posição de impedimento ou não */
		if ((Atacante == Defensor) || (Atacante < Defensor))
		{
			Posicao_legal = 1;
			Repinta_Referencias_Video (5, 10, 10);
		}
		else
		{
			if (Atacante > Defensor)
			{
				Posicao_legal = 2;
				Repinta_Referencias_Video (5, 10, 10);
			}
		}
	}
	else
	{
		/* lado esquerdo do campo */
		distancia = Defensor - Atacante ;
		/* conversão de sinal, para deixar o sinal sempre positivo */
		if (distancia < 0)
			distancia = distancia * (-1);
		/* Verifica se está em posição de impedimento ou não */
		if ((Atacante == Defensor) || (Atacante > Defensor))
		{
			Posicao_legal = 1;
			Repinta_Referencias_Video (5, 10, 10);
		}
		else
		{
			if (Atacante < Defensor)
			{
				Posicao_legal = 2;
				Repinta_Referencias_Video (5, 10, 10);
			}
		}
	}
}
```

QUADRO 7 - Código de cálculo da posição de impedimento

Em seguida, aplica-se a fórmula da fig. 22 para calcular a distâncias entre os pontos em questão.

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

FIGURA 22 – Cálculo distância entre 2 pontos

No quadro 8 encontra-se um trecho de código C que calcula a distância entre 2 pontos no campo de futebol.

```
void Calcula_Distancia (double Atacante, double Defensor)
{
 distancia = 0.00;
/* Calcula a distância entre os 2 pontos selecionados */
 distancia = sqrtl((pow(X_Defensor - X_Atacante,2) + pow(Y_Defensor –
Y_Atacante,2)));
  Repinta_Referencias_Video (6, 10, 10); /* repinta no vídeo */
}
```

QUADRO 8 - Código de cálculo da distância entre 2 pontos

3.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Como forma de validar este trabalho, foi desenvolvido um protótipo aplicando todos os conceitos e técnicas apresentadas. A seguir, é apresentado um estudo de caso para acompanhar a funcionalidade do protótipo:

Dada uma imagem de um lance de uma partida de futebol, deseja-se efetuar a calibração de câmera sobre a mesma, para aplicar o recurso de tira-teima, determinando ou não a posição de impedimento ou calcular a distância entre dois pontos no campo de futebol.

A fig. 23 apresenta a tela principal do protótipo.

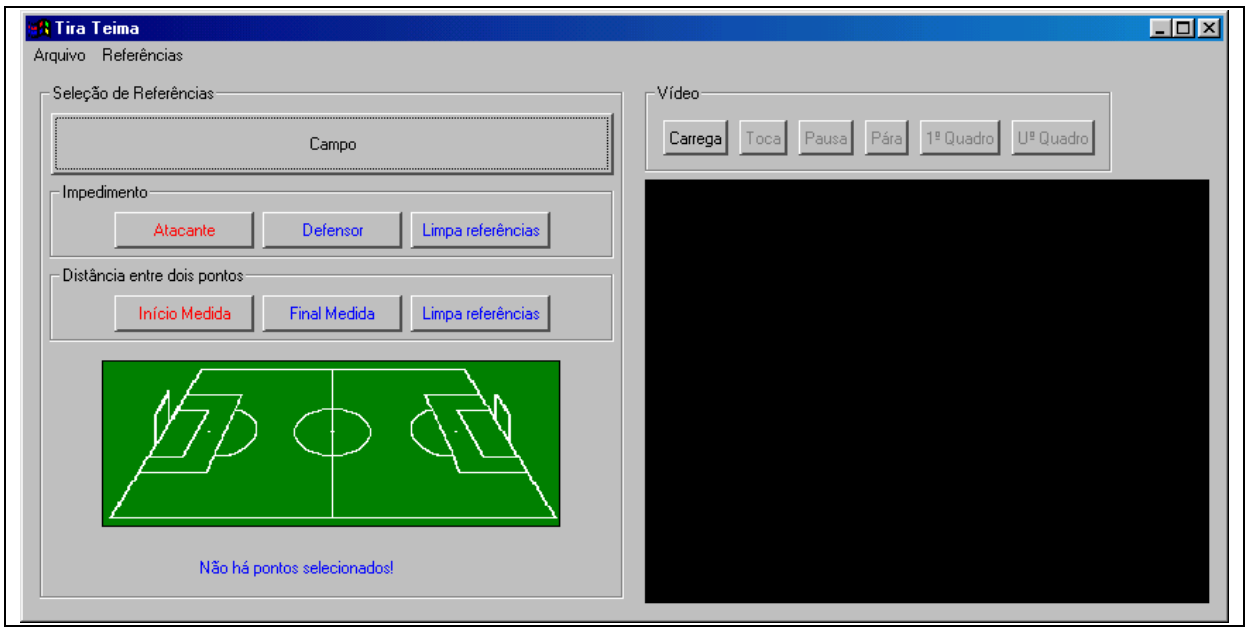


FIGURA 23 – Tela principal do protótipo

Para que o usuário utilize o protótipo, primeiramente ele deverá clicar no botão “Carrega”, para carregar a imagem estática de um lance uma partida de futebol que será analisada. A fig. 24 apresenta a tela que é aberta após clicar no botão “Carrega”.

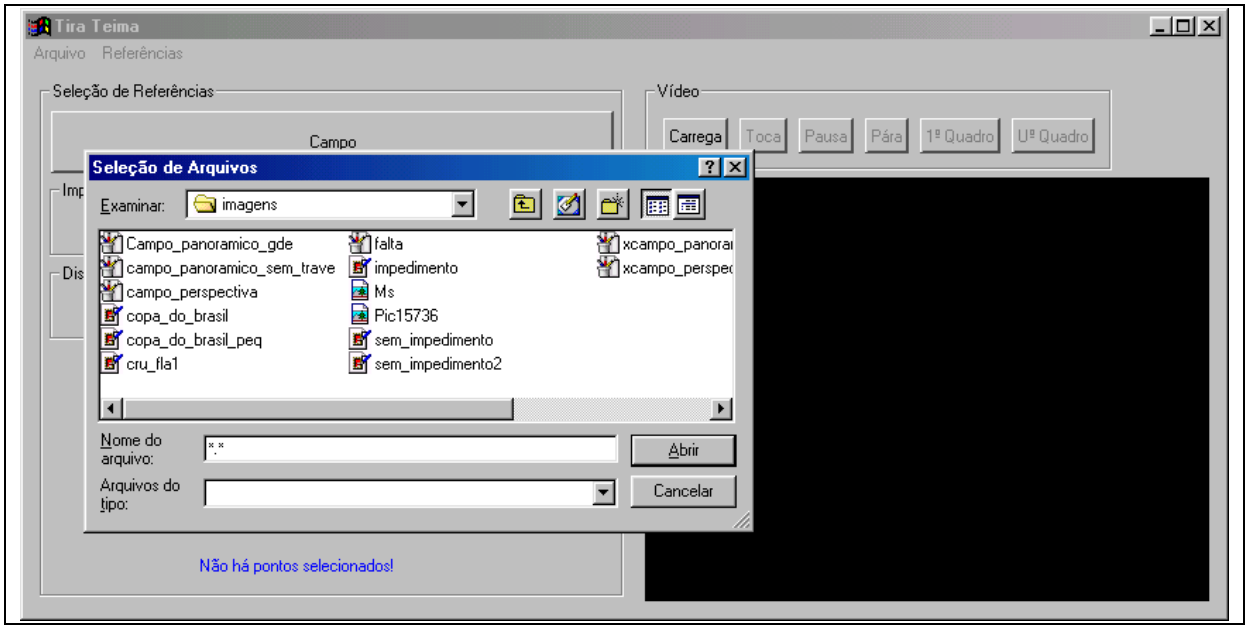


FIGURA 24 – Tela onde é carregada a imagem estática a ser analisada

Após a imagem desejada ser carregada, o protótipo apresenta a tela como é mostrado na fig. 25.

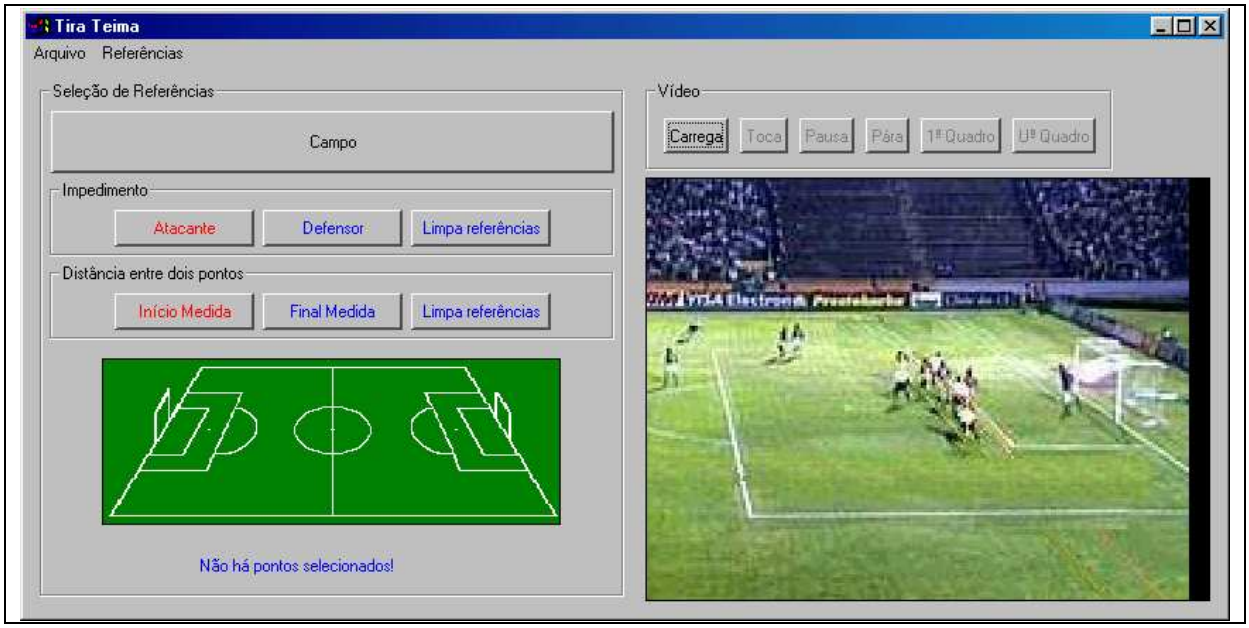


FIGURA 25 – Tela com a imagem estática carregada

Em seguida, o usuário deve clicar no botão “Campo”, e depois ele deve clicar na imagem de vídeo nas interseções das linhas do campo, e em seguida no modelo do campo de futebol na interseção correspondente à selecionada na imagem do lance da partida de futebol. O usuário deve obrigatoriamente selecionar quatro (4) pares na imagem da partida de futebol e os quatro (4) pontos correspondentes no modelo do campo de futebol.

A fig. 26 apresenta todos as possíveis referências de campo que podem ser selecionadas, e a tabela 2 apresenta as suas respectivas coordenadas no sistema de coordenadas de duas dimensões (u,v) e no sistema de coordenadas de três dimensões.

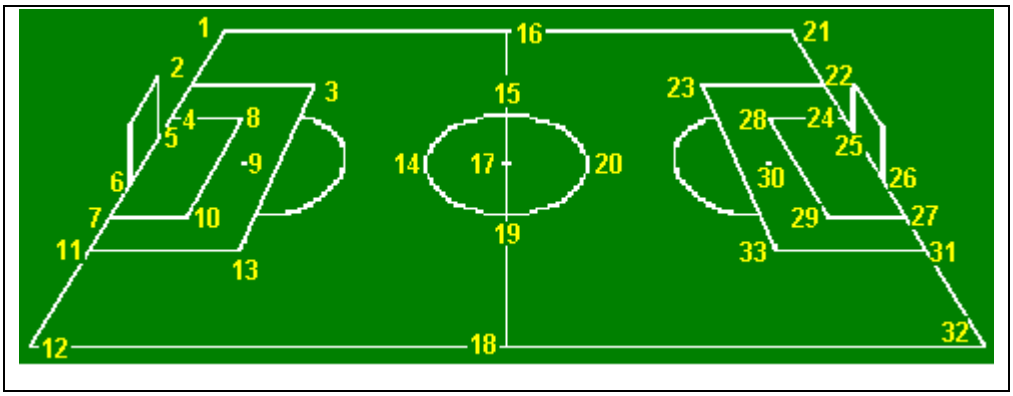


FIGURA 26 – Pontos de referência do campo

Tabela 2 – Descrição dos pontos de referências do campo

Ponto	**Coordenada *u***	**Coordenada *v***	**Coordenada *x***	**Coordenada *y***
1	69	111	0,00	75,00
2	58	93	0,00	57,66
3	98	93	16,50	57,66
4	52	82	0,00	46,60
5	46	73	0,00	41,21
6	37	57	0,00	33,84
7	31	49	0,00	28,34
8	75	82	5,50	46,60
9	74	67	9,15	37,50
10	56	49	5,50	28,34
11	24	38	0,00	21,00
12	4	6	0,00	0,00
13	73	38	16,50	21,00
14	135	67	50,85	37,50
15	162	83	55,00	46,65
16	162	111	55,00	75,00
17	162	75	55,00	37,50
18	162	6	55,00	0,00
19	162	50	55,00	28,35
20	189	67	64,15	37,50
21	256	111	110,00	75,00
22	267	93	110,00	57,66
23	227	93	93,50	57,66
24	274	82	110,00	46,60
25	277	77	110,00	41,21
26	288	59	110,00	33,84
27	294	49	110,00	28,34
28	250	82	104,50	46,60
29	269	49	104,50	28,34
30	249	67	100,85	37,50
31	301	38	110,00	21,00
32	321	6	110,00	0,00
33	252	38	93,50	21,00

Após marcar as referências do campo, é calculada a homografia correspondente à transformação de projeção que teria gerado a imagem do lance da partida de futebol para a transmissão do jogo em televisão. A fig.27 apresenta o sistema de equações apresentado na fig.16 com os valores obtidos com a marcação das referências de campo, e a fig.28 apresenta a matriz dos parâmetros da homografia determinados após a resolução do sistema de equações da fig.27.

A fig. 27 apresenta o sistema de equações da fig. 16, com valores reais obtidos quando da seleção dos pontos 21, 23, 29 e 33 apresentados na tabela 2.

$$\begin{bmatrix} 93.5 & 21 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 119 & 0 & 0 \\ 0 & 0 & 0 & 93.5 & 21 & 1 & 0 & 0 & 0 & 90 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 93.5 & 21 & 1 & -1 & 0 & 0 \\ 93.5 & 57.66 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 147 & 0 \\ 0 & 0 & 0 & 93.5 & 57.66 & 1 & 0 & 0 & 0 & 0 & -32 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 93.5 & 57.66 & 1 & 0 & -1 & 0 \\ 104.5 & 28.34 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -63 \\ 0 & 0 & 0 & 104.5 & 28.34 & 1 & 0 & 0 & 0 & 0 & 0 & 40 \\ 0 & 0 & 0 & 0 & 0 & 0 & 104.5 & 28.34 & 1 & 0 & 0 & -1 \\ 110 & 75 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 110 & 75 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 110 & 75 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 34.5 \\ 54 \\ 1 \end{bmatrix}$$

FIGURA 27 – Sistemas de equações com valores dos pontos de referência do campo

$$H = \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 9.36 \\ -1.22 \\ -903.30 \\ 0.20 \\ 1.69 \\ -94.91 \\ 0.01 \\ 0.01 \\ -0.97 \\ 0.45 \\ 0.67 \\ 0.64 \end{bmatrix}$$

FIGURA 28 – Parâmetros da homografia calculados

A tela do protótipo apresenta indicações dos pontos selecionados, conforme apresenta a fig. 29.

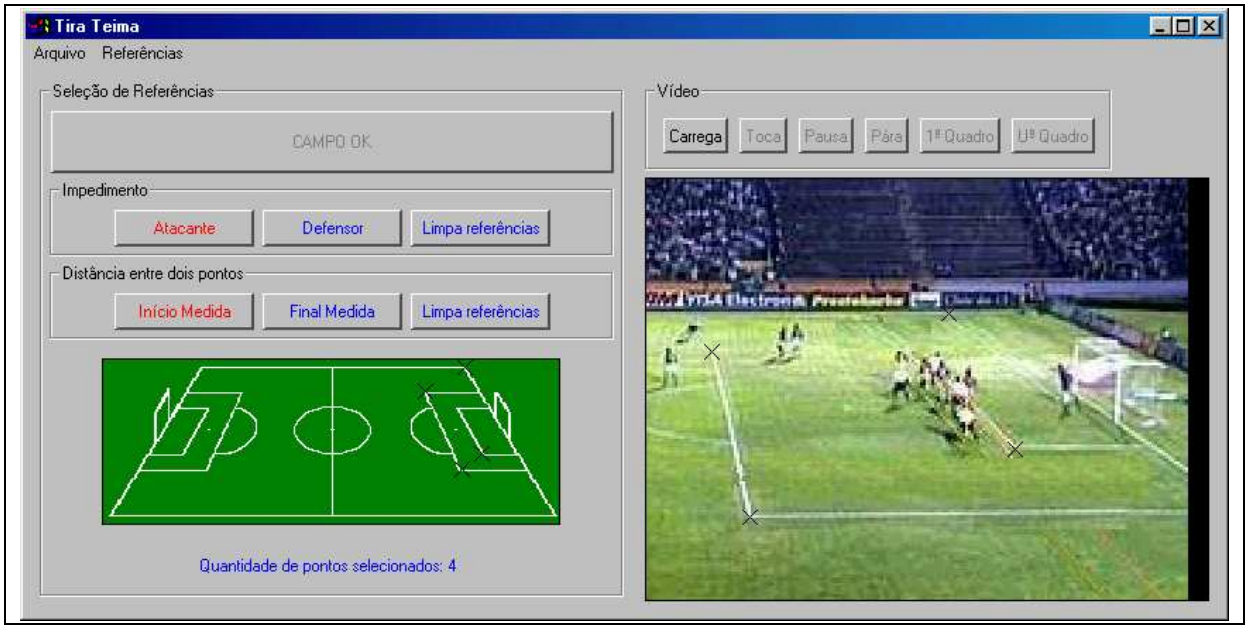


FIGURA 29 – Tela com as referências de campo assinaladas

Após o cálculo da homografia para a cena apresentada na imagem do lance da partida de futebol, o usuário pode determinar o impedimento ou a distância entre 2 pontos. Para calcular o impedimento, o usuário deve clicar no botão “Atacante” e depois clicar na imagem da partida de futebol, na base do atacante (lembra-se que os pontos utilizados devem estar na superfície do campo, com coordenada $z = 0$). Após indicar a posição do atacante, deve-se clicar no botão “Defensor” e de forma idêntica indicar a base do defensor na imagem da partida de futebol. A fig. 30 apresenta a tela do protótipo com as seleções de impedimento efetuadas e com o resultado calculado, havendo a indicação da posição de impedimento ou não.

FIGURA 30 – Tela com o impedimento calculado

Caso o usuário queira calcular a medida entre dois (2) pontos na mesma cena (já que a câmera está calibrada), ele deve clicar no botão “Limpar Referências” da fig. 30. Com as referências limpas, o usuário deve clicar no botão “Início Medida” e indicar na imagem de vídeo o ponto inicial para a medida. Após indicar o ponto na imagem, o usuário deve clicar no botão “Final Medida” e também deve indicar na imagem de vídeo o ponto final para que seja efetuada a medida. A fig.31 apresenta a tela do protótipo com o cálculo da distância da bola até a barreira na cobrança de uma falta, sobre uma imagem de outra partida de futebol.

FIGURA 31 – Tela apresentando a distância calculada

3.4 RESULTADOS E DISCUSSÃO

Um dos maiores problemas enfrentados na realização deste trabalho foi encontrar um método eficiente para efetuar a calibração de câmera (homografia). Como o material disponível para estudo não apresentava uma descrição clara, buscou-se um método que já havia sido aplicado. Ainda assim, ocorreram problemas de compreensão do modelo matemático, que foi solucionado em contato com um pesquisador da área e autor de algumas referências utilizadas (SZENBERG, 1998, 2001).

Outro problema encontrado refere-se à imprecisão numérica apresentada pelo protótipo, pois como foram utilizados apenas quatro (4) pares de pontos de referência do campo para calcular a calibração da câmera que gerou a imagem, trabalha-se com pontos muito dispersos, gerando um modelo de câmera que pode apresentar uma certa variação de resultados. Apesar de haver uma certa imprecisão, a homografia pôde ser calculada. Os resultados obtidos podem sofrer pequenas variações de acordo com os pontos selecionados de referências. É importante ressaltar que, caso o usuário selecione uma referência na imagem estática da partida de futebol que não seja coplanar, e no modelo de campo esta seja informada como uma referência coplanar, haverá uma calibração de câmera incorreta, gerando resultados errados, pois o sistema cartesiano de coordenadas calculado para a câmera

gera um modelo de campo totalmente “distorcido”, apresentando resultados dos pontos selecionados de forma errônea.

A tabela 3 apresenta um comparativo dos resultados obtidos pelo protótipo em relação a outros aplicativos, utilizando como pontos de referência no protótipo os dois cantos da grande área, um canto da pequena área e um *corner*:

Tabela 3 – Posição de impedimento entre o protótipo Tirateima contra outros aplicativos

Imagem	**Protótipo Tirateima**	**Comparativo com:**	**Imprecisão (cm)**
Impedimento	1,86 m	Rede Globo: 1,44 m	22 cm
Impedimento	1,30 m	Juiz Virtual: 1,00 m	30 cm

A tabela 4 apresenta uma repetição da verificação apresentada na tabela 2, utilizando como pontos de referência no protótipo um canto da grande área, um canto da pequena área, um pé da trave e a linha de fundo da grande área:

Tabela 4 – Posição de impedimento entre o protótipo Tirateima contra outros aplicativos

Imagem	**Protótipo Tirateima**	**Comparativo com:**	**Imprecisão (cm)**
Impedimento	0,92 m	Rede Globo: 1,44 m	52 cm
Impedimento	0,14 m	Juiz Virtual: 1,00 m	86 cm

Outra verificação efetuada para analisar os resultados do protótipo está apresentado na tabela 5, onde são comparados os resultados do protótipo com relação às medidas reais de um campo de futebol.

Tabela 5 –Pontos do mundo calculados entre o protótipo Tirateima contra as medidas reais

Local comparado	**Protótipo Tirateima**	**Medidas reais**	**Imprecisão (cm)**
Pequena área	5,45 m	5,50 m	5 cm
Grande área	16,52 m	16,50 m	2 cm
Marca do pênalti	11,20 m	11 m	20 cm

A tabela 6 apresenta os resultados obtidos pelo módulo de medida entre 2 pontos, comparando o resultado obtido com as medidas reais de um campo de futebol.

Tabela 6 –Distâncias calculadas pelo protótipo Tirateima em relação às medidas reais

Local comparado	**Protótipo Tirateima**	**Medidas reais**	**Imprecisão (cm)**
Semicírculo área	9,56 m	9,15 m	41 cm
Bola até barreira	9,95 m	9,15 m	80 cm
Marca do pênalti	12,41 m	11 m	141 cm

Os resultados obtidos para validação do protótipo são bastante positivos, havendo a possibilidade de obter resultados mais precisos, com a utilização de mais pontos de referência, gerando um modelo mais completo.

4 CONCLUSÕES

A idéia inicial proposta para este trabalho de conclusão de curso era implementar um protótipo que pudesse realizar o processo do tira-teima sobre vídeos de jogos de futebol. No decorrer da elaboração do trabalho, surgiram problemas com o uso da biblioteca para a apresentação de vídeos, e tornou-se inviável continuar o protótipo com o uso de vídeos de partidas de futebol. Por este problema, o tira-teima está sendo aplicado apenas em imagens estáticas de lances de partidas de futebol. Até o presente momento, foram implementadas, na forma de um protótipo, todas as etapas para a implementação do tira-teima sobre vídeos, sendo que estaria em aberto apenas implementar uma biblioteca de apresentação de vídeos que retorne informações corretas para o protótipo.

Em relação aos futuros problemas que se apresentam frente à continuidade deste trabalho de pesquisa, os resultados obtidos nos testes para o tira-teima foram bastante adequados, apesar da atual margem de erro considerável, pois está se trabalhando com a quantidade mínima de pontos necessários para efetuar a calibração de câmeras. Sabe-se que a quantidade referências interfere diretamente na precisão do resultado, e pode-se afirmar que quanto mais pontos de referência utilizados, melhores resultados obter-se-á.

4.1 EXTENSÕES

Como sugestões para trabalhos futuros, além da implementação completa do tira-teima aplicado em vídeos de jogos de futebol, que era o objetivo inicial da proposta do trabalho, sugere-se ampliar a implementação para diversos outros esportes, tais como o tênis, o basquete e o vôlei. Esta tarefa requer uma adaptação do método apresentado, no sentido de contemplar as particularidades geométricas das quadras diferentes de um campo de futebol. Em resumo, há a necessidade de implementar parâmetros diferentes para que seja gerada a matriz de projeção baseada nos vídeos destas modalidades.

Sugere-se que em trabalhos futuros seja implementada a ampliação do número de referências utilizadas para efetuar a calibração de câmeras. Desta forma, ter-se-ia um sistema de equações com mais equações do que incógnitas. Com tal sistema, pode-se usar o método dos mínimos quadrados para minimizar o erro encontrando-se uma solução mais precisa para a homografia.

Outra sugestão para trabalhos futuros é a implementação de um método de reconhecimento automático das linhas do campo de futebol e dos objetos de uma partida de futebol sobre vídeos de futebol, gerando uma calibração de câmera dinâmica, sem interação do usuário, obtendo o recurso do tira-teima em tempo real O trabalho de Koser (2003) implementa a primeira etapa deste processo, o reconhecimento automático das linhas do campo. A partir do levantamento automático destas linhas, pode-se calcular a operação de homografia nos moldes apresentados no presente trabalho.

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 uma vez calibrada a câmera, é possível aplicar uma grande variedade de efeitos sobre as imagens, tais como gerar modelos da cena para um ambiente virtual ou ainda aplicar técnicas de publicidade virtual.

REFERÊNCIAS BIBLIOGRÁFICAS

ANTON, Howard; RORRES, Chris. **Álgebra linear com aplicações**. Porto Alegre: Bookman, 2001. 572 p.

CBF – CONFEDERAÇÃO BRASILEIRA DE FUTEBOL. **Regras oficiais de futebol.** Rio de Janeiro: Sprint, 2000. 42 p.

CHANG, Shi-kuo. ***Principles of pictorial information systems design*.** Englewood: Prentice-Hall International Editions, 1989.

IMPA. **Juiz virtual**, Rio de Janeiro, [2003?]. Disponível em: <http://www.visgraf.impa.br/juizvirtual>. Acesso em: 27 nov. 2003.

KOSER, Everton Elvio. **Reconhecimento automático de linhas de campos de futebol em arquivos de vídeo para publicidade virtual.** 2003. 59 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau.

KRUGLINSKI, David J.; WINGO, Scot.; SHEPHERD, George. **Programming Microsoft Visual C++**. Redmond: Microsoft, 1998.

LAWSON, Terry. **Álgebra linear.** São Paulo: Edgard Blücher, 1997. 348 p.

LEINECKER, Richard C.; NYE, Jamie. **Visual C++**: ferramentas poderosas. São Paulo: Berkeley Brasil, 1995.

MARTINO, José Mario de. **Sistemas de informações gráficas**, Campinas, [2003?]. Disponível em: <http://www.dca.fee.unicamp.br/~martino/disciplinas/ea978/na8.pdf>. Acesso em: 15 out. 2003.

NEWMANN, William M.; SPROULL, Robert F. **Principles of interactive computer graphics**. Singapore: McGraw-Hill, 1979. 541 p.

PRESS, William H.; FLANNERY, Brian P, et.al. **Numerical recipes in C**. Cambridge: Cambridge University Press, 1988. 735 p.

PUC-RIO DEPARTAMENTO DE INFORMÁTICA. **Estrutura de dados**, Rio de Janeiro, [2003?]. Disponível em: <http://www.inf.puc-rio.br/~inf1620/material.html>. Acesso em: 18 out. 2003.

QUATRANI, Terry. **Modelagem visual com Rational Rose 2000 e UML**. Rio de Janeiro: Ciência Moderna, 2001. 206 p.

STEINBRUCH, Alfredo; WINTERLE, Paulo. **Álgebra linear**. São Paulo: Mc-Graw Hill, 1987. 583 p.

SZENBERG, 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.

SZENBERG, Flávio et al. Image-based modeling using a two-step camera calibration method. In: Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens, X., 1998, **Anais.** Rio de Janeiro: IMPA, 1988. p. 388-395.

TECGraf. **Ferramentas de desenvolvimento de programas**, Rio de Janeiro, [2003?]. Disponível em: <http://www.tecgraf.puc-rio.br/f_prodp/ferr_c.htm>. Acesso em: 19 out. 2003.

WANGENHEIM, Aldo von. **Visão computacional**, Florianópolis, [2003?]. Disponível em: <http://www.inf.ufsc.br/~visao>. Acesso em: 14 out. 2003.

WATT, Alan. **Fundamentals o three-dimensional computer graphics**. New York: Addison-Wesley Publishing Company, 1989. 430 p.