UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

ANDRE GRZYBOWSKI ALBANO SILVA

Rastreamento de objetos em tempo real para aplicações em jogos esportivos

São Carlos -2012

Andre Grzybowski Albano Silva

Rastreamento de objetos em tempo real para aplicações em jogos esportivos

Trabalho de Conclusão de
Curso apresentado à Escola de
Engenharia de São Carlos, da
Universidade de São Paulo

Curso de Engenharia Elétrica
com ênfase em Eletrônica

ORIENTADOR: Prof. Dr. Marcelo Andrade da Costa Vieira

São Carlos
2012

Ficha catalográfica preparada pela Seção de Atendimento ao Usuário do Serviço de Biblioteca – EESC/USP

S586r

Silva, Andre Grzybowski Albano
Rastreamento de objetos em tempo real para aplicações em jogos esportivos. / Andre Grzybowski Albano Silva; orientador Marcelo Andrade da Costa Vieira. São Carlos, 2012.

Monografia (Graduação em Engenharia Elétrica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2012.

1. Rastreamento de vídeo. 2. Visão computacional. 3. Transformada de Hough. 4. Processamento digital de imagens. I. Título.

FOLHA DE APROVAÇÃO

Nome: Andre Grzybowski Albano Silva

Título: "Rastreamento de Objetos em Tempo Real para Aplicações em Jogos Esportivos"

Trabalho de Conclusão de Curso defendido e aprovado em 27/06/2012,

com NOTA 8,5 (oito, cinco), pela comissão julgadora:

Marcelo Andrade da Costa Vieira

Prof. Dr. Marcelo Andrade da Costa Vieira (Orientador) - EESC/USP

Valdir Grassi Junior

Prof. Dr. Valdir Grassi Júnior - EESC/USP

Bruno Barufaldi

M. Sc. Bruno Barufaldi - EESC/USP

Homero Schiabel

Prof. Associado Homero Schiabel
Coordenador da CoC-Engenharia Elétrica
EESC/USP

Dedico todo o meu esforço e trabalho em todos esses anos a minha família e amigos.

Agradecimentos

Agradeço à minha família que sempre me apoiou e me possibilitou cursar Engenharia Elétrica na USP. Aos amigos de Cuiabá que compreenderam a minha ausência por longos períodos. Aos meus amigos que fiz aqui no estado de São Paulo e fizeram esses 5 anos valer a pena. Aos membros e ex-membros da Equipe EESC USP Baja SAE pela oportunidade e aprendizado que me proporcionaram. Por fim, aos professores e funcionários da USP São Carlos que sempre me ajudaram. E a todos que de certa forma contribuíram para a conclusão deste trabalho.

"Se, encontrando a Desgraça e o Triunfo, conseguires, tratar da mesma forma a esses dois impostores ... Tua é a Terra com tudo o que existe no mundo, e - o que ainda é muito mais - és um Homem, meu filho!"

(Rudyard Kipling)

SUMÁRIO

RESUMO

Em algumas modalidades esportivas que usam bola e a quadra de jogo é demarcada por linhas, como no caso do tênis e do vôlei, sempre houve certa dificuldade nas marcações de linha pelos juízes. Devido à alta velocidade da bola, em alguns casos, o juíz de linha não consegue definir com precisão se a bola caiu dentro ou fora da quadra, o que pode ocasionar erros durante uma partida. Tendo em vista esse fato, torna-se necessário algum tipo de tecnologia em tempo real que pudesse auxiliar os juízes nesse tipo de marcação a partir da imagem (ou do vídeo) da bola no momento que ela toca a quadra. Já existem e estão em funcionamento, para algumas modalidades esportivas, sistemas de visão computacional que detectam com precisão o caminho percorrido pela bola e definem se ela esteve em contato com as linhas ou não no momento do impacto. Porém, devido ao custo, esse tipo de equipamento só pode ser utilizado por torneios profissionais de grande porte. Dessa forma, esse projeto apresenta um sistema de visão computacional para rastreamento de uma bola de tênis em tempo real usando imagem de vídeo. Foi desenvolvido um sistema que utiliza uma câmera digital de baixo custo, um suporte para essa e um *software* feito em C++ no *Visual Studio* para rastreamento da bola em tempo real e a definição precisa se, no momento do impacto da bola com a quadra, ela tocou na linha ou não. Foram testados três tipos de filtros para o rastreamento: transformada de Hough, conexão de componentes e filtor de Kalman. Resultados mostraram que o método de conexão de componentes apresentou melhor resultado com erro de 8% comparando com a análise visual do vídeo. Desta forma, esse trabalho apresenta um projeto similar aos sistemas comerciais utilizados em jogos esportivos aliando baixo custo e qualidade, voltado para um público de torneios de pequeno porte e usuários recreativos.

Palavras-chave: rastreamento de vídeo, visão computacional, transformada de Hough, processamento digital de imagens.

ABSTRACT

In some sports that uses ball and the court is delimited by lines, such as tennis and volleyball, there has been some difficulties with line calling by the judges. Due to the high speed of the ball, in some cases, the line judge can not define precisely if the ball fell inside or outside the court, which can results in errors during the match. With this in mind, it is necessary some kind of real-time technology that could help judges in this kind of callings using the picture (or vídeo) of the ball when it touches the court. For some sports, computer vision systems that detect the precise path of the ball and determine if it was in contact with the lines or not at the moment of impact already exists and are in operation. However, due to the high price, this type of equipment can only be used by large professional tournaments. Therefore, this project presents a computer vision system for tracking a tennis ball in real time using video image. It has been developed a system that uses a low cost digital camera, a stand to hold it and a software in C++ developed at Visual Studio to track the ball in real-time and the precise definition if, at the impact, it touched or not the line. Three kind of filters has been tested to do the tracking: Hough transform, connected component labeling and Kalman filter. Results shows that connect component labeling was the best method with an error of 8% in comparsion to a visual analysis of the video. So, this Project presents a system similar to commercial systems used in sports combining low cost and quality, aimed at an audience of small tournaments and recreational users.

Key words: video tracking, computer vision, Hough transform, image digital processing.

LISTA DE FIGURAS

LISTA DE TABELAS

INTRODUÇÃO

Atualmente, os esportes têm sofrido um aumento na velocidade em que são praticados. Analisando as modalidades esportivas praticadas com bola, o avanço tecnológico dos equipamentos e da medicina preventiva e corretiva aplicada ao físico dos atletas, têm permitido esse aumento na velocidade e isso influencia de forma direta a velocidade que a bola atinge na partida, além das mudanças abruptas de direção. Todos esses fatores dificultam as marcações de linha pelos juízes devido às suas limitações físicas.

Em alguns esportes, como tênis e críquete, já foram desenvolvidos sistemas de visão computacional, como o Hawk Eye, Auto Ref e EDH Sport (*INTERNATIONAL TENNIS FEDERATION* - ITF, 2010b), que auxiliam nas marcações através do rastreamento da bola para posterior análise do momento que ela toca a quadra, definindo com precisão se ela caiu dentro ou fora da quadra. Exemplos desses sistemas podem ser vistos na Figura 1.

Figura 1 - Câmera utilizada para filmagem do impacto da bola em uma partida de tênis (ITF,2010b).

Tradicionalmente, o tênis é julgado por seres humanos. Durante o jogo, as decisões a respeito da posição, em relação às linhas da quadra, que a bola tocou o chão são feitas

pelos árbitros de linha e o juíz de cadeira. No entanto, devido à velocidade da bola, há ocasiões em que existe incerteza por parte dos juízes na marcação do ponto, pois não conseguem definir com precisão o ponto de impacto da bola. Isso gera uma série de polêmicas e problemas durante o jogo que podem definir um resultado de uma partida erroneamente. Isso ocorre porque o impacto entre a bola e a superfície acontece em alta velocidade com duração de menos de 0,01 segundo (ITF, 2010a) e, além disso, normalmente há mudanças bruscas na trajetória da bola durante uma partida, o que dificulta a marcação dos pontos pelos juízes.

Avanços na área de visão computacional nos últimos anos resultaram no desenvolvimento de vários sistema de determinação automática da posição de impacto da bola na quadra esportiva como o de Yan, Christman e Kittler citados posteriormente neste trabalho. Entretanto, os custos desses sistemas são muito altos o que faz com que estejam disponíveis apenas em torneios de alto nível com grandes recursos financeiros. É possível então, perceber, a necessidade de desenvolvimento de um sistema que alie qualidade e baixo custo nesse segmento.

1. OBJETIVO

Este projeto tem como objetivo o desenvolvimento de um sistema de visão computacional de baixo custo para aplicação em jogos esportivos capaz de detectar com precisão se a bola esteve em contato ou não com a linha da quadra. Futuramente poderá ser utilizado em torneios profissionais de pequeno porte e amadores.

O objetivo desse trabalho de conclusão é apresentar um sistema de visão computacional com as seguintes características:

- Utilização de filmadoras de pequeno porte (câmera digital; *point and shoot*) podendo o sistema ser conectado diretamente a qualquer câmera já existente excluindo a necessidade de compra de uma;
- Utilização de *hardware* de pequeno porte (*desktops* e *notebooks*) através do uso mínimo de processamento priorizando *frames* em que há a necessidade de marcação de linha;
- Baixo custo e com nível equivalente à sistemas já existentes comercialmente, mas muito mais caros;
- Uso básico da computação gráfica apenas para geração da imagem final (bola em contato com a quadra ou com a linha) excluindo o trajeto da bola como em alguns sistemas existentes.

Apesar de ser voltado para qualquer modalidade esportiva que utiliza bola e quadra, o sistema apresentado nesse trabalho, por ser um projeto inicial, foi desenvolvido apenas com base em um jogo de tênis, de tal forma a validar e testar os métodos e o sistema para um posterior desenvolvimento de um sistema generalizado para outras esportes.

2. REVISÃO BIBLIOGRÁFICA

Nesse capítulo, busca-se mostrar a fundamentação teória que foi utilizada no desenvolvimento do sistema de visão computacional proposto.

Para localizar automaticamente a quadra nas imagens capturadas, rastrear a bola, detectar o ponto de impacto e determinar se houve o contato ou não entre a bola e a linha deve-se capturar as imagens de vídeo, segmentar e detectar os objetos de interesse quadro-a-quadro, rastreá-los e interpretar os resultados da forma conveniente. Para isso, utilizaram-se filtros, detectores, operações morfológicas, transformadas e técnicas de processamento de imagens e visão computacional que serão explicados a seguir.

2.1. ESPAÇO DE CORES

Um espaço de cor é um abstrato modelo matemático descrevendo a forma que as cores podem ser representadas como uma sequência de números (GONZALEZ; WOODS, 2008).

Ao se adicionar uma função de mapeamento entre o modelo de cores e um espaço de referência, resulta-se em uma gama de cores e, em combinação com o modelo, define um novo espaço.

Espaços de cores podem ser definidos sem o uso de um modelo de cores. Esses espaços são o efeito de um dado conjunto de nomes e número que são definidos pela existência de um correspondente conjunto de amostra de cores físicas.

2.1.1. SISTEMA RGB

Um sistema de cores RGB é definido pela intensidade de vermelho (*red*), verde (*green*) e azul (*blue*) que compõem cada cor. O propósito principal desse sistema é a reprodução de cores em dispositivos eletrônicos como computadores, monitores, câmeras digitais e projetores.

Sua representação é baseada na atribuição de 3 *bytes* para cada *pixel* em uma imagem. Cada byte representa o nível de intensidade de cada umas das cores RGB, sendo atribuídos valores de 0 a 255 para cada pixel em cada canal de cor.

Esse modelo pode ser representado por um cubo como pode ser visto Figura 2.

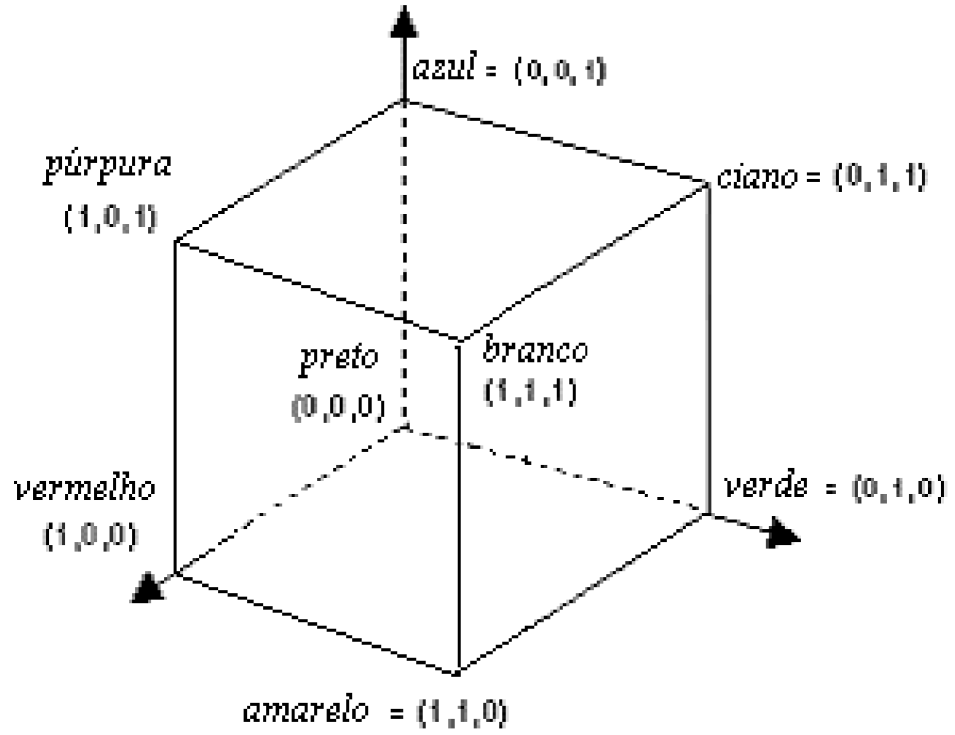

Figura 2 - Cubo representando o espaço de cor RGB (GONZALEZ; WOODS, 2008).

2.1.2. SISTEMA HSV

HSV é a abreviatura para o sistema de cores formado pelas componentes matiz (*hue*), saturação (*saturation*) e valor (*value*). Esse sistema de cores define o espaço de cor conforme descrito abaixo, utilizando seus três parâmetros:

- Matiz (tonalidade): Verifica o tipo de cor, abrangendo todas as cores do espectro, desde o vermelho até o violeta, mais o magenta. Atinge valores de 0 a 360, mas para algumas aplicações, esse valor é normalizado de 0 a 100%;
- Saturação: Também chamado de "pureza". Quanto menor esse valor, mais com tom de cinza aparecerá a imagem. Quanto maior o valor, mais "pura" é a imagem. Atinge valores de 0 a 100%;

- Valor (brilho): Define o brilho da cor. Atinge valores de 0 a 100%.

Considerando os valores percentuais dos canais RGB, os canais HSV podem ser obtidos através dos canais RGB utilizando as equações 1, 2 e 3 (JACK,2007) a seguir.

$$H = \begin{cases} 60 * \frac{G-B}{\max-\min}, se\ \max = R\ e\ G \geq B \\ 60 * \frac{G-B}{max-min} + 360, se\ \max = R\ e\ G < B \\ 60 * \frac{B-R}{max-min} + 120, se\ max = G \\ 60 * \frac{R-G}{max-min} + 240, se\ max = B \end{cases} \quad (1)$$

$$S = \begin{cases} \frac{max-min}{max}, se\ max \neq 0 \\ 0, se\ max = 0 \end{cases} \quad (2)$$

$$V = max \quad (3)$$

O modelo HSV pode ser representado por um cone com base hexagonal como mostrado na Figura 3.

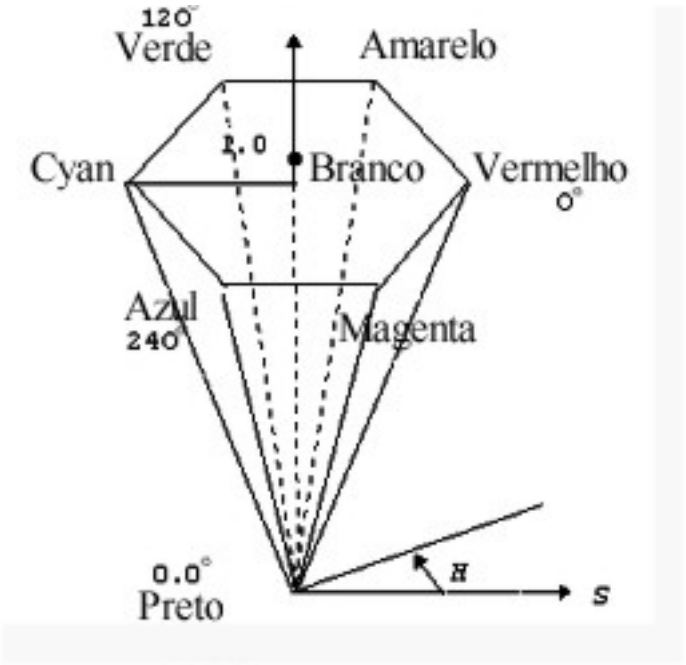

Figura 3 - Cone hexagonal representando o espaço de cor HSV (JACK,2007).

2.2. EQUALIZAÇÃO DE HISTOGRAMA

A equalização do histograma de uma imagem tem por objetivo aumentar o contraste geral dessa espalhando a distribuição de níveis de cinza (VIEIRA, 2010).

A equalização pode ser obtida através da equação 4.

$$q = \max\{0, arredondamento\left(\frac{\sum_{j=0}^{k} n_j}{I}\right) - 1\}, g \geq k \geq 0 \quad (4)$$

Em que *g* corresponde aos níveis de cinza da imagem original e *q* aos níveis de cinza da imagem equalizada. Na Figura 4, pode ser visto um exemplo de equalização.

Figura 4 - Exemplo de imagem original (à esquerda) e equalizada (à direita).

2.3. SUAVIZAÇÃO

O processo de suavização tem por objetivo eliminar ruídos e suavizar a imagem através de um filtro espacial passa-baixa. Pode ser realizado utilizando um filtro no domínio do espaço conhecido como média da vizinhança (GONZALEZ; WOODS, 2008). Um exemplo de filtro que realiza a média de uma vizinhança de um pixel pode ser visto na equação 5.

$$A = \frac{1}{5}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (5)$$

Na Figura 5, a seguir, pode ser observado o resultado de uma suavização por um filtro passa-baixa.

Figura 5 - Exemplo de suavização GONZALEZ; WOODS, 2008).

2.4. OPERAÇÕES MORFOLÓGICAS

A morfologia de imagens tem utilidade como técnica de análise e processamento de geometrias estruturais em uma imagem, baseada na teoria dos conjuntos, topologia e funções randômicas.

No caso de imagens digitais, o processamento de imagem morfológico consiste em uma série de operações que transformam as imagens de acordo com a suas características.

A morfologia pode funcionar como uma ferramenta para a extração de componentes de imagens que sejam úteis na representação e descrição da forma de uma região, como fronteiras e esqueletos. Também é muito utilizada na etapa de pré e pós-processamento, como filtragem morfológica, afinamento, etc (PRATT, 1991).

Originalmente desenvolvida para imagens binárias, a morfologia matemática foi estendida para imagens e funções na escala de cinza.

2.4.1. EROSÃO

A erosão de imagem binária A pelo elemento estruturante B é definido pela equação 6.

$$A \ominus B = \{z \in E | B_z \subseteq A\} \quad (6)$$

Na equação 7, pode ser visto Bz que é a translação de B pelo vetor.

$$B_z = \{b + z | b \in B\}, \forall z \in E \quad (7)$$

Quando o elemento estruturante B tem um centro e esse está localizado na origem de E, então a erosão de A por B pode ser entendido como o lugar geométrico dos pontos alcançados pelo centro de B quando B se move dentro de A. Um exemplo, que pode ser visto na Figura 6, é a erosão de um quadrado de lado 10, centrado na origem, por um disco de raio 2, também centrado na origem, também é um quadrado de lado 6 centrado na origem.

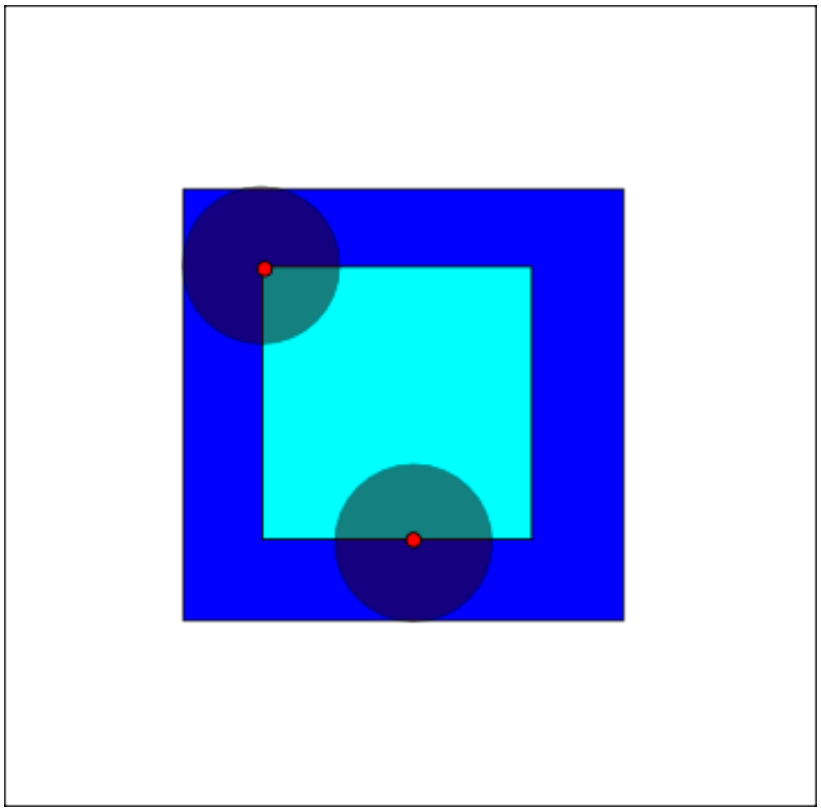

Figura 6 - Exemplo de erosão aplicada a um quadrado.

O resultado dessa operação em uma imagem pode ser entendido como uma “redução” no seu tamanho.

2.4.2. DILATAÇÃO

A dilatação de A pelo elemento estruturante B é definida pela equação 8.

$$A \oplus B = \cup A_b, b \in B \tag{8}$$

A dilatação é comutativa, também dada pela equação 11.

$$A \oplus B = B \oplus A = \cup B_a, a \in A \tag{9}$$

Se B tem um centro na origem, então a dilatação de A por B pode ser entendida como o lugar geométrico dos pontos cobertos pelo B quando o centro de B se move em A. Como pode ser visto na Figura 7, a dilatação do quadrado de lado 10 pelo disco de raio 2 é um quadrado de lado 14, com cantos arredondados, centrada na origem. O raio dos cantos arredondados é 2.

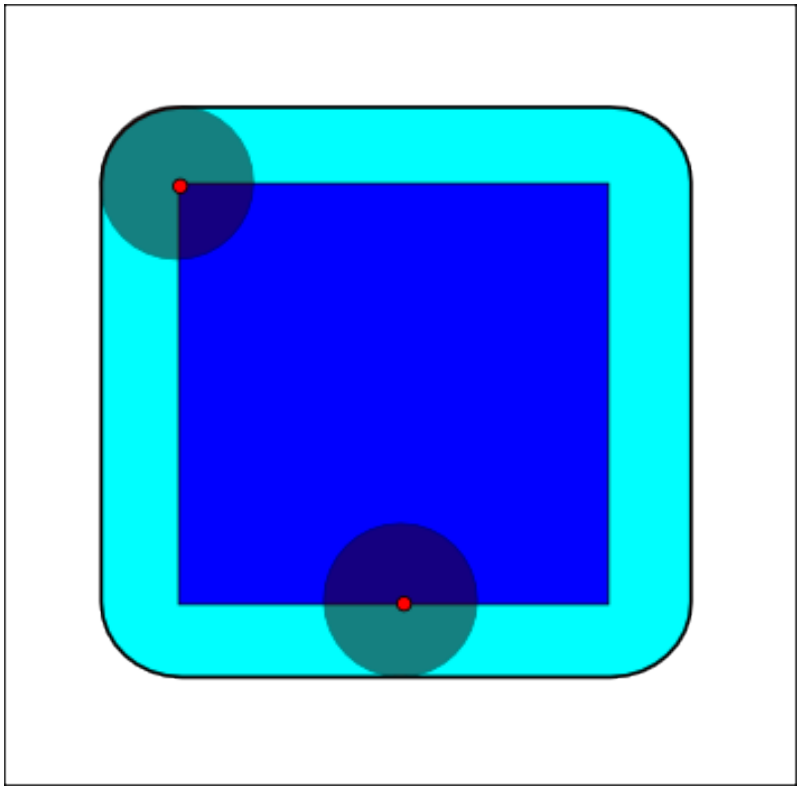

Figura 7 - Exemplo de dilatação no mesmo quadrado.

O resultado dessa operação em uma imagem pode ser entendido como um “aumento” no seu tamanho.

2.4.3. FECHAMENTO

O fechamento de A por um elemento estruturante B é obtido pela dilatação de A por B, seguido de uma erosão da imagem resultante por B como descrito na equação 10.

$$A \cdot B = (A \oplus B) \ominus B \tag{10}$$

O fechamento remove todos os pixels onde o ajuste do elemento estruturante não está dentro da imagem e enche todos os espaços onde o elemento estruturante não iria se ajustar na imagem.

Um exemplo com o quadrado de lado 10 e o disco de raio 2 pode ser visto na Figura 8.

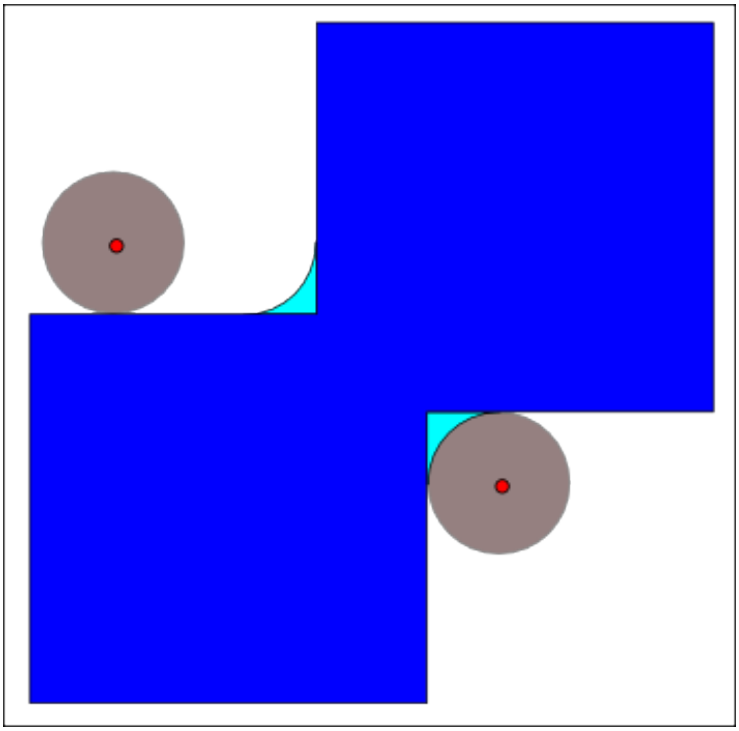

Figura 8 - Exemplo de operação de fechamento.

2.4.4. ABERTURA

A abertura de A por B é obtida pela erosão de A pelo elemento estruturante B, seguido pela dilatação da imagem resultante por B como pode ser descrito pela equação 11.

$$A \circ B = (A \ominus B) \oplus B \tag{11}$$

No caso do quadrado de lado 10, como pode ser visto na Figura 9, e um disco de raio 2 como o elemento estruturante, a abertura é um quadrado de lado 10 com cantos arredondados, onde o raio de canto é 2.

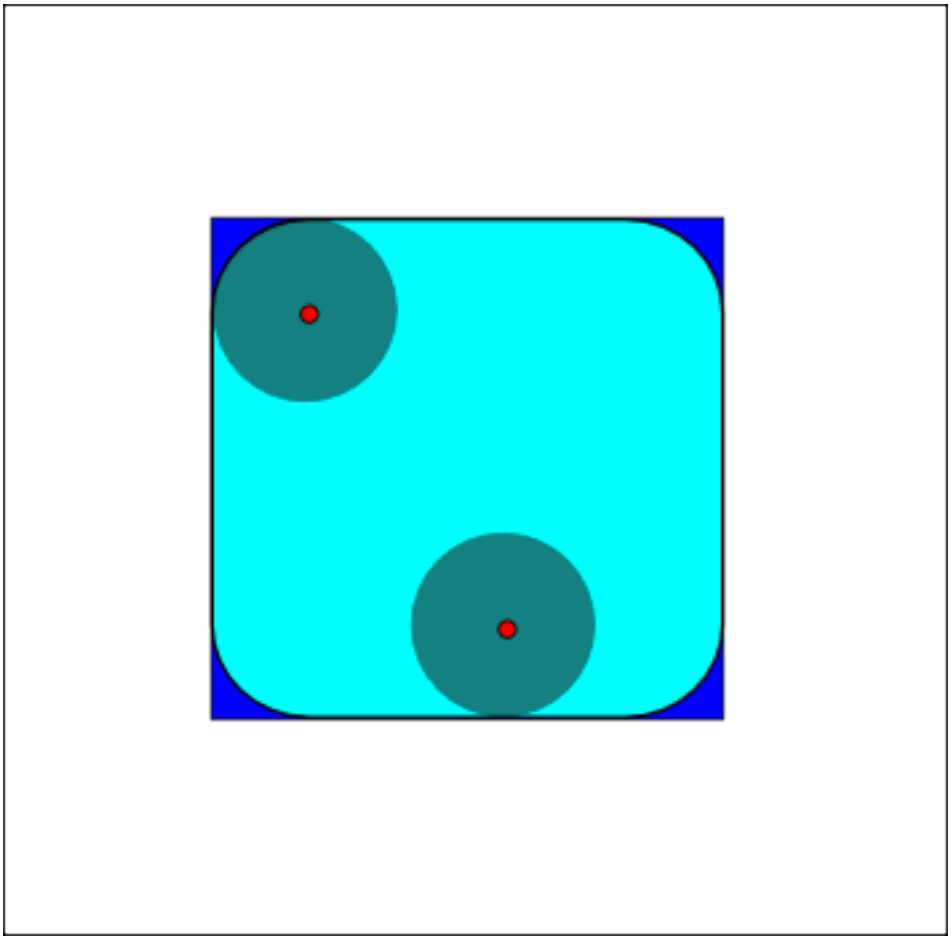

Figura 9 - Exemplo de uma operação de abertura.

A abertura suaviza o contorno de uma imagem e elimina pequenos pontos, ou seja, elimina ruídos da imagem.

2.5. SEGMENTAÇÃO

A segmentação é o processo de agrupamento de partes de uma imagem em conjuntos que tenham em comum alguma característica.

Uma imagem geralmente pode ser segmentada de acordo com duas propriedades dos seus níveis de cinza: descontinuidades ou similaridades, gerando, respectivamente, dois tipos básicos de segmentos: fronteiras ou regiões.

2.5.1. DESCONTINUIDADES

Uma descontinuidade em uma imagem pode ser detectada de três formas:

1. Um ponto isolado;
2. Linhas;
3. Bordas de objetos.

O primeiro caso é o mais simples. Ocorre se um ponto isolado possui um nível de cinza de valor muito diferente da vizinhança (alto ou baixo).

Na detecção de linhas deve ser detectado pontos semelhantes e testá-los com o intuito de verificar se estão em uma linha. O uso de máscaras, como a mostrada na equação 12, pode ajudar a detectar linhas horizontais, verticais e diagonais.

$$A = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad (12)$$

Por fim, uma borda de um objeto é o limite entre duas regiões com níveis de cinza distintos. Para sua detecção, trabalha-se com a magnitude da primeira e segunda derivada da distribuição de nível de cinza na imagem (CASTLEMAN, 1996).

A primeira derivada pode ser usada na detecção da presença de uma borda em uma imagem. Já a segunda derivada pode ser usada para determinar se um ponto da borda está no lado escuro ou claro. Dessa forma, utilizam-se operadores gradientes como Roberts, Sobel e Prewitt para detecção da presença de borda e operadores laplacianos para

detecção do lado em que o ponto se encontra. Na prática, o laplaciano é pouco utilizado por ser muito complexo e sensível a ruídos.

Apesar do alto custo computacional, um dos melhores métodos de detecção de bordas é o detector de Canny. Ele é um operador gaussiano de primeira derivada que suaviza os ruídos e localiza as bordas. Seu funcionamento se dá a partir dos seguintes estágios (WANGENHEIM, 2000):

1. A imagem é suavizada por um filtro gaussiano;
2. Ocorre a diferenciação, ou seja, é calculada a direção do gradiente em cada ponto. Portanto, serão geradas cristas onde há borda;
3. São suprimidos os pontos de não-máximo;
4. Ocorre a limiarização das bordas com histerese, ou seja, as bordas “fracas”, se próximas de uma borda “forte”, são conectadas a ela.

A Figura 10 mostra um exemplo de aplicação do filtro de Canny em uma imagem.

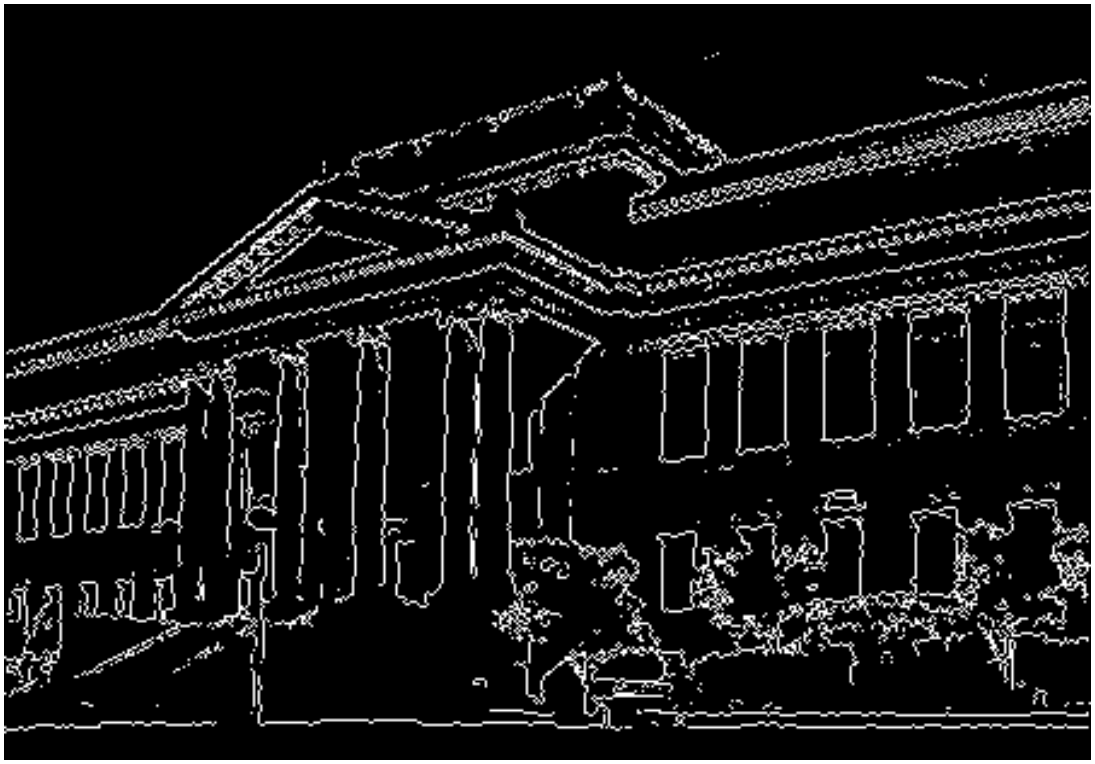

Figura 10 - Exemplo de detector de bordas de Canny (BRADSKI; KAEHLER, 2008).

2.5.2. TRANSFORMADA DE HOUGH

A transformada de Hough é utilizada para a detecção de curvas que possam ser descritas de forma paramétrica (reta, círculo, etc). O conceito principal da transformada está em definir um mapeamento entre o espaço de imagem e o espaço de parâmetros. Cada borda de uma imagem é transformada pelo mapeamento para determinar células no espaço de parâmetros, indicadas pelas primitivas definidas através do ponto analisado. Essas células são incrementadas e indicarão, no final do processo, através da máxima local do acumulador, quais os parâmetros correspondem à forma especificada.

O algoritmo de Hough requer um acumulador de dimensão igual ao número de parâmetros desconhecidos na equação da família de curvas que são buscadas. Por exemplo, achar segmentos de linhas usando a equação da reta requer achar dois parâmetros para cada segmento: a e b.

Assim, usando uma matriz acumuladora A, o procedimento de Hough examina cada ponto e calcula os parâmetros da curva especificada que passa pelo ponto. Caso esteja analisando uma imagem que não foi pré-processada com algoritmo de detecção de bordas, será examinado o ponto e sua vizinhança na imagem, para determinar se há evidência de extremidade nele. Somente se isso acontecer será realizado o cálculo dos parâmetros (BRADSKI; KAEHLER, 2008).

Quando todos pixels tiverem sido processados, é procurado no acumulador A os maiores valores. Eles indicam os parâmetros de prováveis linhas na imagem. A Figura 11, abaixo, mostra um exemplo de aplicação da transformada de Hough em uma imagem.

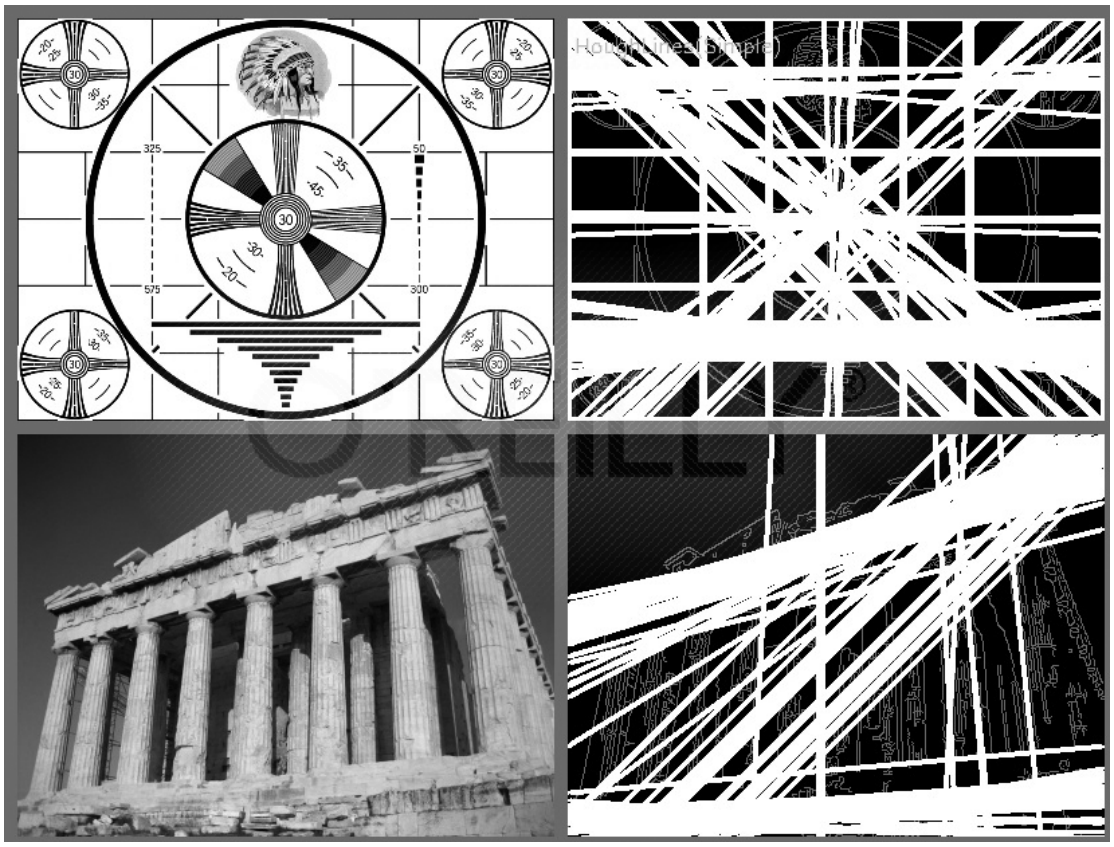

Figura 11 - Exemplo de transformada de Hough (BRADSKI; KAEHLER, 2008).

2.5.3. LIMIARIZAÇÃO

O processo de binarização, que ocorre com uma limiarização, tem como característica a divisão da imagem em dois níveis de cinza apenas: branco (máximo) e preto (mínimo). Esse processo leva em conta um nível de cinza médio, também chamado de *threshold*, em que todos os pontos com nível de cinza superior a ele são considerados brancos e todos abaixo são considerados pretos.

A escolha do nível de cinza médio pode ocorrer de duas formas:

1. Análise visual do histograma da imagem;
2. De forma automática, utilizando de um processo interativo em que são testados vários *thresholds* até a obtenção de um resultado satisfatório.

Esse processo automático pode ser realizado através do método de Otsu, em que a imagem é tratada como uma função densidade de probabilidade (WANGENHEIM, 2010).

Considerando duas classes, C1 com nível de cinza menor que o *threshold t* e C2 com nível de cinza maior que *t*, o objetivo do método é maximizar a variância entre as classes em relação à variância global, conforme a equação 13 a seguir.

$$\sigma^2 = P(m1 - mg)^2 + P(m2 - mg)^2 \quad (13)$$

Na equação 13, *P* é a probabilidade de um ponto pertencer a certa classe, *m1* e *m2* são as intensidades médias dos pontos de cada classe, *mg* é a intensidade media de todos pontos e *σ* é a variância global (ALDO, 2000).

A limiarização também pode ocorrer em multinível. Nesse caso, em vez de uma binarização, tem-se mais de um *threshold* e a imagem é dividida em mais de dois níveis (o tanto que for desejado). A Figura 12, abaixo, mostra um exemplo de binariazação em uma imagem.

Figura 12 - Exemplo de imagem binária.

2.5.4. DIFERENÇA DE QUADROS

O método mais simples de diferença do fundo é subtrair um quadro do outro (possivelmente alguns quadros depois) e depois marcar qualquer diferença que seja grande o suficiente como primeiro plano. Esse processo tende a detectar as bordas de objetos em movimento. Por simplicidade, vamos dizer que temos dois quadros da sequência de imagens. Se um quadro tiver a imagem antiga em escala de cinza e outro for a imagem atual também em escala de cinza, então se subtrairmos um do outro teremos as diferenças do primeiro plano como resultado (BRADSKY; KAEHLER, 2008).

Devido ao fato dos valores dos pontos exibirem ruídos e flutuações, devem-se ignorar valores muito baixos e marcar o resto como importante, ou seja, realizar uma binarização na imagem.

Apesar da binarização da imagem, ainda sobrar um pouco de ruído devido à pequenos movimentos no fundo (sujeira, vento, etc). A filtragem destes pode ser realizada através da operação morfológica de erosão e/ou utilizando conexão de componentes.

Para imagens coloridas, o processo pode ser feito para cada canal separadamente e depois serem recombinados.

Na Figura 13, abaixo, pode ser visto um exemplo de diferença de quadros. No caso, foi feita a diferença entre quadros conseguintes de um vídeo resultando no contorno do objeto em movimento no vídeo.

Figura 13 - Exemplo de diferença de quadros.

2.5.5. CONEXÃO DE COMPONENTES

Conexão de componentes é usada em visão computacional para detectar regiões conectadas em imagens digitais preto e branco, embora as imagens coloridas ou dados com maior dimensão também possam ser processados. Quando integrado num sistema de reconhecimento de imagem ou interface humano-computador, conexão de componentes pode operar em uma variedade de informações. A extração de regiões é geralmente realizada na imagem binária resultante a partir de um processo de limiarização.

Um gráfico, contendo vértices e arestas de ligação, é construído a partir de dados de entrada relevantes. Os vértices contêm as informações exigidas pela heurística de comparação, enquanto as bordas indicam vizinhos conectados. Um algoritmo percorre o gráfico, identificando os vértices com base na conectividade e valores relativos de seus vizinhos. A conectividade é determinada pela média; gráficos de imagem, por exemplo, podem ser ligados ou por conectividade 4 ou por conectividade 8.

Dois pontos são ditos ligados por conectividade 4 se possuem níveis de cinza similares e adjacentes por bordas. Já para serem ligados por conectividade 8 eles devem, além de terem níveis de cinza similares, ser adjacentes por vértice (VIEIRA, 2010). Um exemplo pode ser visto na Figura 14.

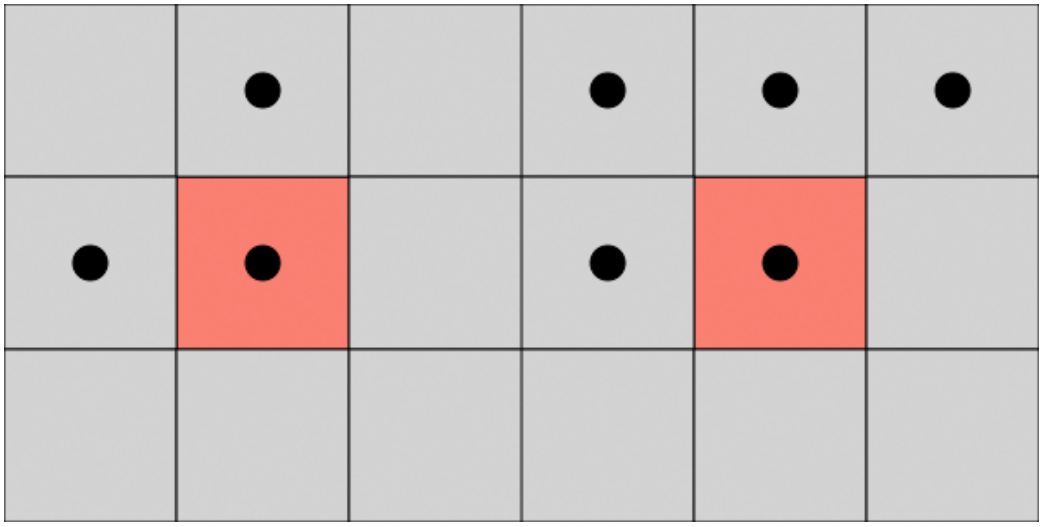

Figura 14 - Conectividade 4 e 8.

Na Figura 14, acima, os pontos pretos tem todos o mesmo nível de cinza. O ponto destacado em vermelho à esquerda compartilha as bordas entre os pixels acima e ao lado esquerdo dele. Por isso, que o ponto em vermelho tem conectividade 4 com os outros

pontos. Já em relação ao ponto em vermelho a direita, ele compartilha vértice com o ponto a nordeste. Dessa forma, é dito que o ponto tem conectividade 8 com os outros. É evidente da definição que alguns pontos poderão ter conectividade 4 ou 8.

Após a etapa de rotulagem, o gráfico pode ser dividido em subgrupos, de modo que a informação original possa ser recuperada e processada.

2.6. FILTRO DE KALMAN

Fusão de dados usando um filtro de Kalman pode ajudar no rastreamento de objetos em vídeos com bastante ruído. O rastreamento de objetos é um problema dinâmico, usando dados de sensores e imagens de câmera que sempre sofrem de ruído. Isto às vezes pode ser reduzido pelo uso de câmeras de alta qualidade e sensores, mas nunca pode ser eliminado, por isso muitas vezes é desejável usar um método de redução de ruído (WELCH; BISHOP, 2001).

A natureza iterativa preditor-corretor do filtro de Kalman pode ser útil, porque a cada passo discreto de tempo apenas uma informação sobre a variável de estado deve ser considerada. Este processo é repetido, considerando-se uma informação diferente a cada passo de tempo. Todos os dados medidos são acumulados ao longo do tempo e ajudam na previsão do estado. Os vídeos também podem ser pré-processados, talvez usando uma técnica de segmentação, para reduzir o cálculo e, portanto, a latência.

A idéia básica por trás do filtro de Kalman é que, sob algumas suposições, será possível, dado um histórico de medições de um sistema, construir um modelo para o estado do sistema que minimize os erros entre a predição e o valor real da variável. Uma grande vantagem do filtro de Kalman é o fato dele iterativamente atualizar o modelo de estado de um sistema e manter apenas o modelo, sem necessidade de guardas as medidas, para a próxima iteração. Isso simplifica muito as implicações computacionais deste método.

Há três suposições importantes necessários na construção teórica do filtro de Kalman: (1) o sistema a ser modelado é linear, (2) o ruído que as medições estão sujeitas é "branco", e (3) esse também é Gaussiano. O primeiro pressuposto significa que o estado do sistema no tempo *k* pode ser modelado como uma matriz multiplicada pelo estado no tempo de *k-1*. O pressuposto adicional que o ruído é tanto branco e Gaussiano significa que esse não é correlacionado com o tempo e que a sua amplitude podem ser modelada com precisão usando apenas uma média e uma covariância (ou seja, o ruído é completamente

descrito pelo seu primeiro e segundo momentos). Embora estes pressupostos podem parecer restritivo, eles ,na verdade, se aplicam a um grande conjunto de circunstâncias.

O filtro de Kalman é, dadas as três hipóteses, a melhor maneira de combinar dados de fontes diferentes ou da mesma fonte em momentos diferentes (BRADSKY; KAEHLER, 2008). Ele se inicia com o que é sabido, obtém-se novas informações, e então é decidido mudar o modelo com base no erro entre a predição e a medida, utilizando uma combinação ponderada do modelo velho e do novo.

2.6.1. DESENVOLVIMENTO MATEMÁTICO

O filtro de Kalman assume um estado de espaço linear em que a transição de estado é da forma da equação 14 em que *x* é o estado do sistema em determinado tempo *k*, *u* é a variável de controle e *w* é o ruído do processo. As matrizes *F* e *B* definem o modelo do sistema ao longo do tempo.

$$x_k = F_k x_{k-1} + B_k u_k + w_k \tag{14}$$

O modelo de transição de estado é aplicado ao estado anterior. O ruído do processo, representado por *w*, é da transição de estado que não é modelada e assume-se que é um ruído branco Gaussiano com covariância *Q* expressa na equação 15.

$$Q_k \delta(k-j) = E[w_k w_j^T] \tag{15}$$

A entrada de controle, representada por *u*, é mapeada no espaço estimado por *B*, o modelo da entrada de controle.

A observação (medida) *z* é assumida que é da forma da equação 16.

$$z_k = H_k x_k + v_k \tag{16}$$

Na equação 16, *H* é o modelo de observação e *v* é o ruído de observação em um momento *k* assumindo ruído branco Gaussiano com covariância *R* expressa pela equação 17.

$$R_k \delta(k-j) = E[v_k v_j^T] \quad (17)$$

As equação 18 e 19 representam as equações para o modelo de predição.

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k \quad (18)$$

$$\hat{P}_{k|k-1} = F_k \hat{P}_{k-1|k-1} F_k^T + Q_k \quad (19)$$

Já as equações 20, 21 e 22 representam as equações para o modelo de atualização.

$$K_k = \hat{P}_{k|k-1} H_k^T (H_k \hat{P}_{k|k-1} H_k^T + R_k)^{-1} \quad (20)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - H_k \hat{x}_{k|k-1}) \quad (21)$$

$$\hat{P}_{k|k} = (I - K_k H_k) \hat{P}_{k|k-1} \quad (22)$$

A Figura 15, a seguir, mostra como funciona o filtro através da aplicação das equações mostradas acima.

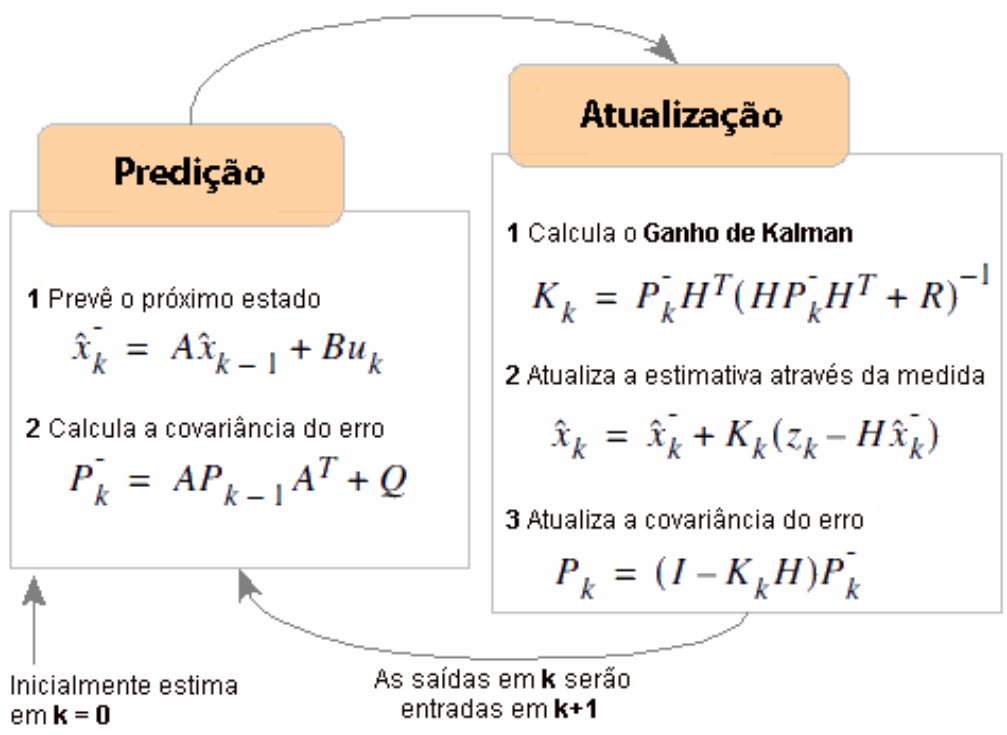

Figura 15 - Esquemático de funcionamento do filtro de Kalman.

3. MATERIAIS E MÉTODOS

Esse trabalho propõe um sistema de visão computacional para reconhecimento e rastreamento de esportes com bola – no caso, aplicado ao tênis para fins de testes e exemplo - com o objetivo de auxiliar nas marcações de linha determinando se a bola está dentro ou fora. A idéia é possibilitar um sistema de baixo custo que possa estar acessível a torneios de baixo nível e até mesmo ao público recreativo. Neste contexto, o trabalho aborda os problemas e soluções desde a aquisição de imagens, segmentação, reconhecimento, rastreamento e interpretação dos resultados adquiridos.

Tendo em vista o baixo custo, os principais pilares desse projeto são a análise, escolha e desenvolvimento de filtros e métodos que compensem a falta de equipamentos de alta precisão e exatidão no rastreamento. Dessa forma, esse projeto divide-se em três etapas: aquisição das imagens, segmentação, rastreamento e solução para o problema proposto. O sistema segue o diagrama mostrado na Figura 16 a seguir.

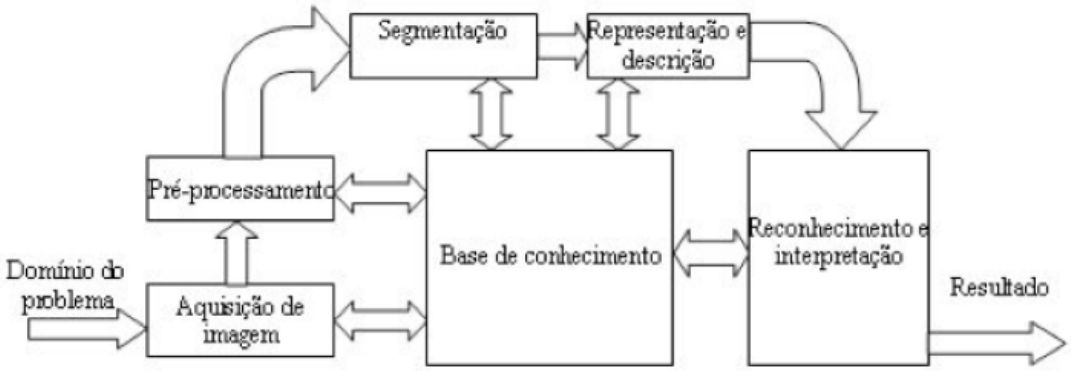

Figura 16 - Diagrama de um sistema de visão computacional.

Levando em conta a literatura encontrada a respeito de projetos de rastreamento de objetos semelhantes, pode se observar a complexidade de alguns quantos às técnicas e métodos para rastreamento. Por exemplo, analisando o trabalho de Yan, Christmas e Kittler (2005), pode ser observado que foi utilizado algoritmos de grande complexidade. Dessa forma, alguns dos sistemas não puderam ter suas alternativas testadas. Por outro lado, os trabalhos remanescentes são, de certa forma, complementares resultando em um único algoritmo desenvolvido buscando envolver as melhores técnicas e métodos possíveis.

3.1. DEFINIÇÃO DO SISTEMA

O rastreamento de objetos, como no caso bolas de tênis, permite o auxílio aos seres humanos em diversas tarefas que sobressaem a capacidade humana, incluindo a determinação da posição de bola de tênis no impacto em relação a quadra. O fato de cada vez mais termos empresas investindo nessa área e esportes aderindo a essa tecnologia reforça a idéia de que esse é um setor em pleno desenvolvimento. Além disso, com o avanço da tecnologia ocorrerá um aumento de qualidade e barateamento de produtos nessa área.

Para realizar o rastreamento e análise do vídeo em tempo real são necessários:

1. Aquisição da imagem por meio de uma câmera;
2. Detecção do objeto de interesse;
3. Rastreamento do objeto no vídeo;
4. Análise do objeto e interpretação dos resultados.

Partindo do principio de visão computacional, deve ser definido:

1. Como o objeto será representado;
2. Como será feita a detecção;
3. Como será feito o rastreamento.

3.1.1. REPRESENTAÇÃO DE OBJETOS

Um objeto, nas imagens que ele será rastreado, pode ser definido por qualquer característica que seja interessante para seu reconhecimento e análise depois (GONZALEZ; WOODS, 2008). Dessa forma, temos:

1. Ponto: O objeto de interesse pode ser representado por um ponto. Seja o centro da bola, das linhas ou até mesmo um conjunto de pontos;
2. Forma geométrica: Para o caso da bola de tênis, a forma geométrica buscada é o círculo. Já para as linhas da quadra, da forma trapezoidal;
3. Contorno: Tanto a bola como as linhas podem ser caracterizadas pelo seu contorno resultando nas formas geométricas de cada um;

4. Cor: As linhas da quadra são sempre brancas podendo ser representada pelo nível de cinza máximo ou próximo desse. Já a bola, apesar de não ser branca, também é clara podendo ser identificada pelo seu nível de cinza.

3.1.2. DETECCÇÃO DE OBJETOS

A detecção dos objetos é, assim como as outras fases, parte crucial no rastreamento do objeto de interesse. Essa acontecerá, em geral, de acordo com a forma de representação adotada. Por exemplo, a bola pode ser detectada de acordo com a diferença de cor dela com o resto do ambiente, ou seja, por limiaridade (GONZALEZ; WOODS, 2008). Dessa forma, teremos a nosso dispor as seguintes formas de detecção:

1. Cor – Limiaridade/Binarização: Como explicado, a bola poderá ser detectada por sua cor diferente. Assim como as linhas da quadra que possuem valor de cinza máximo possível.
2. Forma geométrica – Detectores de borda/Transformada de Hough: As bordas dos objetos podem ser detectadas para posterior detecção da forma geométrica (linha ou círculo) com a transformada de Hough.
3. Contorno – Detectores de borda: Assim como explicado acima, será utilizado para possibilitar a detecção da forma e seus parâmetros.

3.1.3. RASTREAMENTO DOS OBJETOS

O objetivo do rastreamento é detectar em cada quadro do filme um mesmo objeto e segui-lo ao longo das imagens. Ele pode ser realizado a partir de correspondência individual em que em cada quadro o objeto é detectado de acordo com os método mostrado acima e correlacionados depois. O rastreamento também pode ser realizado em conjunto, sendo estimado a posição do objeto através de detecção em quadros anteriores. Entre os métodos de rastreamento temos o filtro de Kalman, método de Lucas-Kanade, algoritmo de Horn-Schunck e filtro de condensação (BRADSKI; KAEHLER, 2008). Na Figura 17, pode ser visto um exemplo de rastreamento por fluxo ótico usando o método de Lucas-Kanade.

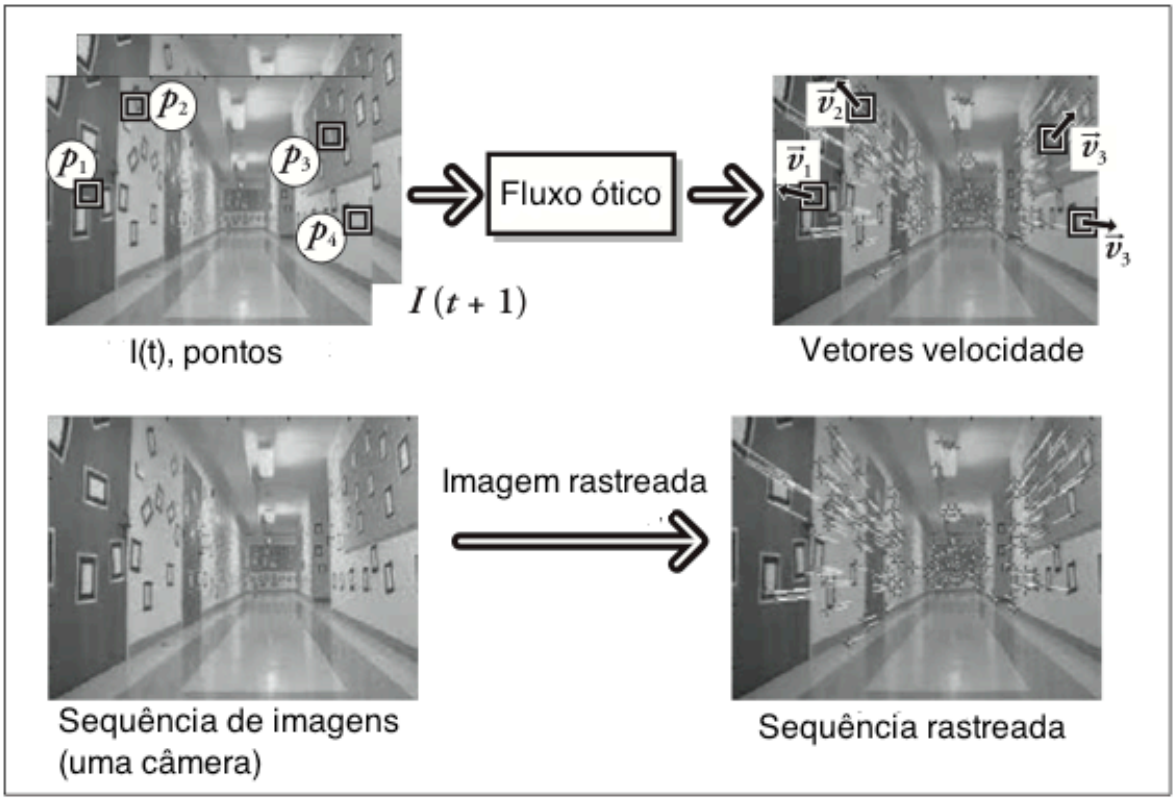

Figura 17 - Exemplo de rastreamento (BRADSKI; KAEHLER, 2008).

A principio, o método utilizado será de correspondência individual por ser o de mais fácil desenvolvimento. Posteriormente, será testado a correspondência em conjunto a fim de resolver problemas de oclusão, deformação e alta velocidade.

3.1.4. PROBLEMAS DA TRAJETÓRIA DA BOLA

O impacto entre uma bola de tênis e uma superfície de corte é complexa. A área de contato entre uma bola em repouso sobre uma superfície é relativamente pequeno. Se uma bola é lançada verticalmente, a área de contato tende a aumentar à medida que o fundo da bola deforma. Em geral, quanto maior sua velocidade na direção vertical quando se atinge a superfície (até certo ponto), mais ela se deforma e maior será a área de contato máximo. Para a grande maioria dos impactos no tênis, a bola está se movendo horizontalmente, bem como na vertical, o que aumenta a complexidade da determinação da localização do verdadeiro impacto (especialmente para árbitros de linha humanos). Devido ao movimento horizontal, a bola escorrega ao longo do solo e, assim, deixa a superfície em uma posição diferente daquela em que primeiro fez contato. A quantidade de mudanças de escorregamento de jogada a jogada dependem da abordagem, ângulo, velocidade, rotação, e do tipo de superfície da quadra, o que torna difícil identificar com precisão o ponto verdadeiro impacto inicial (ITF, 2010a).

Os juízes dos esportes podem usar pistas para o local de impacto. Por exemplo, a bola perturba as partículas da superfície em uma quadra de barro, deixando uma marca na superfície, enquanto na grama, a presença ou ausência de pó de giz é usado às vezes como evidência da localização. Marcas também podem ser vistas em quadras duras. Em todos os casos, no entanto, tais evidências podem ser enganosas. Por exemplo, é possível que partículas de argila não foram atingidas pela bola pode ser perturbado por essas partículas se movendo devido ao impacto da bola, o que resultará em uma maior marca do que a área de contato verdadeiro. Em uma quadra dura, no entanto, a pressão limiar necessária entre a bola e a superfície para deixar uma marca não pode ser gerada até algum tempo após o contato inicial, deixando assim uma marca menor do que a área de contato verdadeiro.

A seguir, na Figura 18, uma exemplificação da deformação da bola com o impacto e velocidade.

Figura 18 - Exemplo da bola comprimindo e deformando (ITF, 2010a).

3.1.4.1. PROBLEMAS DO AMBIENTE

É evidente que o ambiente não é ideal. Dessa forma, podem e irão ocorrer ruídos nas imagens capturadas, problemas de luminosidade – sob ou sobre exposição – e imperfeições das imagens dos objetos de interesses a serem detectados e rastreados como cor não uniforme e imperfeições de geometria tanto na bola como nas linhas da quadra.

Portanto, torna-se necessário a utilização de filtros que eliminem os ruídos das imagens, equalização de histograma a fim de corrigir os problemas de luminosidade e filtros e operações que possibilitem a transformação dos objetos o mais próximo possível de suas formas geométricas e uniformidade de cores.

Além disso, a segmentação por cor fica comprometida quando a bola está sobre a linha. Torna-se complexo estabelecer um limiar que filtre a bola, porém não considere a linha ou vice-versa. Levando em conta que detecção por borda depende da imagem estar

na escala de cinza, torna-se necessário buscar outra forma de segmentação. Portanto, o método de diferença de quadros é interessante para realizar esse tipo de segmentação.

3.1.4.2. DINÂMICA DA BOLA

O movimento da bola de tênis pode ser decomposto como um movimento na direção vertical de queda livre e um movimento na direção horizontal de velocidade constante. Mesmo que a direção entre a gravidade e velocidade na direção vertical mude logo após o impacto da bola com o solo ocorra fricção nesse momento, o que realmente interessa ao projeto é o momento exatamente antes ao impacto em que a posição é a mesma do impacto, mas sem sofrer novas acelerações. Dessa forma, o sistema pode ser considerado linear nos instantes de interesse. O ruído na medida é presumido por ser da forma de uma distribuição Gaussiana, então o uso de filtro de predição e atualização, como o de Kalman, pode ser utilizado no auxílio do rastreamento da bola (WU, 2008). O sistema pode ser expressado pela equação 23

$$\begin{bmatrix} x_{t+1} \\ \dot{x}_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} + \begin{bmatrix} T^2 \\ 2T \end{bmatrix} a \qquad (23)$$

Na equação 23, *x* é a posição da bola, *T* é a taxa de amostragem, que dependerá de quantos quadros por segundo a câmera suporta (no caso, 30 quadros por segundo), a derivada de *x* é a velocidade da bola e *a* é a aceleração que a bola sofre.

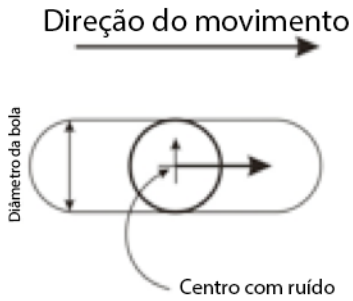

Figura 19 - Representação do movimento da bola com ruído.

Além disso, conforme citado e mostrado anteriormente, a bola tem um problema de deformação quando em impacto ou em alta velocidade. Isso aliado a limitação de 30 quadros por segundo da câmera, contribui para a aquisição de imagens da bola que se confundam com um borrão. De tal maneira, torna-se complexa a segmentação da bola utilizando técnicas comuns como limiarização, detector de bordas e transformada de Hough. Um exemplo pode ser visto na Figura 19 acima em que a posição real da bola (círculo no centro) não representa o que é capturado pela câmera (a elipse em volta).

Utilizando-se mais de um quadro, pode ser realizada a diferença entre os dois e o resultado será a diferença de movimento da bola. Ainda assim, o resultado não permite inferir a posição da bola nem sua dimensão. Faz-se necessário então a utilização de um método que, a partir de várias medições, possibilite encontrar o componente principal do movimento da bola e filtre o borrão.

Assume-se que o borrão aconteça apenas na direção do movimento, mas não na perpendicular. Pode-se utilizar, então, uma ferramenta como a PCA (análise do componente principal) para encontrar o componente principal do movimento da bola.

3.1.4.3. PROBLEMAS DE OCLUSÃO

Um outro problema que comumente ocorre é a oclusão. Devido a outros objetos que aparecem na cena - como jogadores, raquetes e outros objetos ou animais que podem aparecer devido ao fato do esporte ser praticado em um ambiente aberto – a bola pode, em alguns instantes, não estar ao alcance da câmera. Este problema pode ser resolvido de duas maneiras:

1. Uso de duas ou mais câmeras de tal forma que a bola nunca esteja fora do alcance da câmera;
2. Uso de algum algoritmo que ajude a estimar a posição da bola em momentos de oclusão.

O problema da primeira opção é que, mesmo que sejam usadas várias câmeras para captura das imagens, ainda assim é possível que existam objetos suficientes para que ocorra a oclusão da bola. Além disso, essa opção acarretaria em um considerável aumento de custo.

Um exemplo de problemas de oclusão por jogadores pode ser visto na Figura 20.

Figura 20 - O jogador pode estar a frente ou acima da bola a escondendo dependendo do ponto de vista da câmera.

Dessa forma, a melhor opção para esse projeto seria o uso de algoritmos com função de estimar a posição da bola combinado ao fato de que ajudaria a prever a dinâmica como explicado anteriormente.

3.1.5. TRABALHOS E PROJETOS RELACIONADOS

Dentre vários projetos e trabalhos na área de visão computacional, encontrou-se vários artigos e dissertações relacionados ao rastreamento de bolas de tênis ou mesmo objetos no geral.

Nibert e Spencer (2008) implementaram um sistema em *OpenCV* para realizar o rastreamento em tempo real de bolas de tênis. A imagem era capturada por uma câmera ligada ao computador por *FireWire* e utilizavam a cor e forma geométrica para detecção. A correspondência foi individual sem nenhum método de predição. Apesar de terem realizados estudos sobre filtro de Kalman, não chegaram a implementar.

Yan, Christmas e Kittler (2005) desenvolveram um sistema de rastreamento de bola de tênis com apenas uma câmera de baixa qualidade. Seu sistema detecta em cada *frame* os candidatos a bolas e depois desenvolveram um algoritmo de simulação Monte Carlo considerando a dinâmica de bola linear. Conseguiram, assim, determinar possíveis caminhos e, recursivamente, otimizar esses caminhos resultando no caminho real da bola rastreando com precisão a bola mesmo com vídeo de baixa qualidade e oclusões

randômicas. O sistema implementado detectou também com sucesso o impacto da bola com o piso, filtrando os impactos com as raquetes dos jogadores. Na Figura 21, abaixo, pode visto o resultado do algoritmo com os pontos referentes à bola e a sua trajetória.

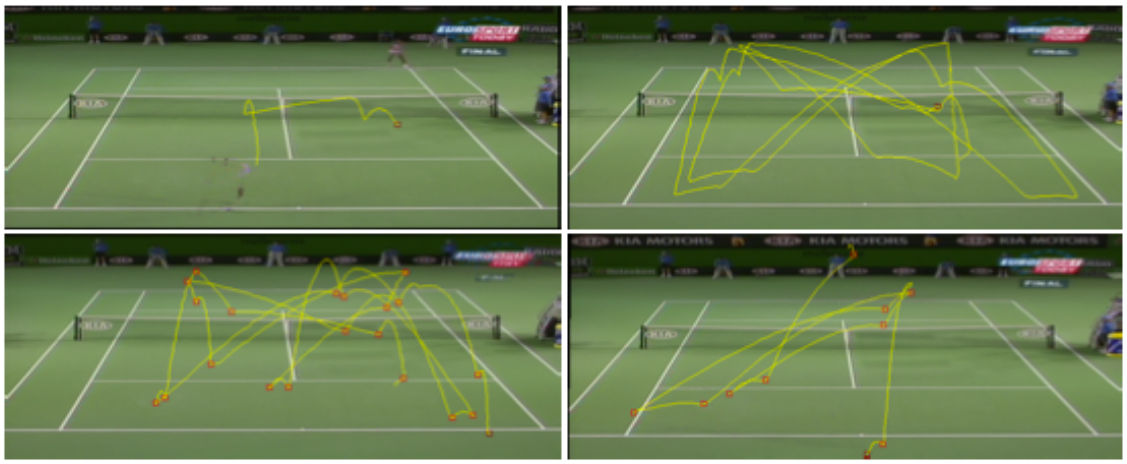

Figura 21 - Algoritmo de Yan aplicado em um vídeo de baixa qualidade (YAN; CHRISTMAS; KITTLER, 2005).

Wu (2009) desenvolveu um sistema de rastreamento de bola de tênis e detecção do ponto de impacto de baixo custo utilizando apenas uma câmera filmadora capturando a 50 quadros por segundo. Ele utilizou de segmentação por cores para encontrar as linhas da quadra e segmentação por diferença para encontrar a bola de tênis. Seu projeto utilizou de PCA para encontrar a posição da bola e filtra o borrão e filtro de Kalman para estimar a posição da bola em casos de oclusão e falha na segmentação quando a bola está em cima da linha.

3.2. SISTEMA E MATERIAIS

O sistema desenvolvido utiliza uma câmera *point and shoot*, vista na Figura 22, ligada a um computador para a aquisição de imagens em tempo real e posterior tratamento com o intuito de rastrear o movimento da bola e se o impacto com o chão (quique) ocorreu dentro ou fora da quadra.

Figura 22 - Câmera *point and shoot* utilizada.

Como a iluminação afeta a aquisição de imagens, as imagens foram adquiridas apenas em ambientes abertos no período diurno de tal forma a otimizar a abertura da câmera e seu tempo de exposição.

O computador utilizado para os experimentos desse projeto possui processador Intel Core 2 Duo com 2,53 GHz de clock e 4 MB de memória cache, memória (RAM) de 4 GB 1067 MHz DDR3, placa gráfica NVIDIA 9400M 256 MB e disco rígido de 250 GB rodando o sistema operacional Windows XP Professional 64 bits. As imagens foram capturadas por um câmera *point and shoot* com resolução de 720p (1280x720) e com uma taxa de 30 quadros por segundo.

Para processamento do vídeo, este foi convertido para o formato AVI, com uma taxa de dados de 10,73 Mbits por segundo e utilizando o *codec* cvid. As imagens de cada quadro foram extraídas pelo algoritmo no formato JPEG de mesma resolução no espaço de cor RGB com 24 bits (8 bits por canal).

O algoritmo foi desenvolvido utilizando o software *Microsoft Visual Studio 2010 Ultimate* juntamente com a biblioteca *OpenCV 2.2*, que disponibiliza várias funções comumente relacionadas a visão computacional como de filtragem, segmentação, detecção e técnicas de rastreamento como filtro de Kalman. Essa biblioteca permitiu o foco no desenvolvimento do sistema e em sua qualidade em detrimento do desenvolvimento de funções comuns do ambiente de visão computacional. Apesar do ambiente de

desenvolvimento escolhido, esse pode ser facilmente transferido para qualquer outro sistema operacional devido à portabilidade da biblioteca *OpenCV* e da linguagem de programação *C++*.

Na Figura 23, pode ser visto um esquemático de como foi feita a captura das imagens que geraram os vídeos testes para o algoritmo. Uma fita adesiva branca foi utilizada para simular a linha da quadra enquanto um suporte garantiu a vista superior para a câmera e a sua estabilidade.

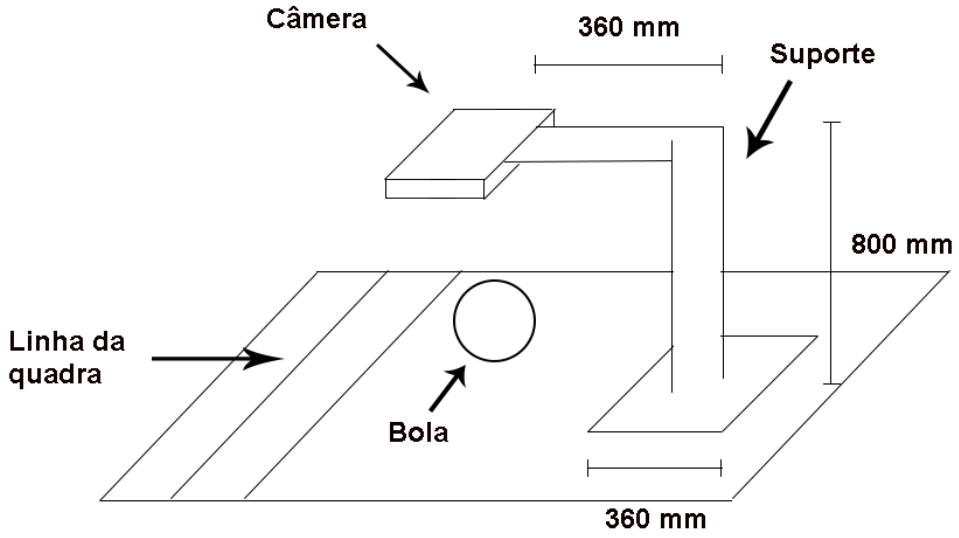

Figura 23 - Esquema de captura de imagens.

Na Figura 24, a seguir, pode ser visto o sistema montado com o suporte, a câmera e a fita adesiva simulando a linha da quadra.

Figura 24 - Sistema de captura de imagens montado.

3.3. CONDIÇÕES DE CONTORNO E HIPÓTESES

Como pode ser visto nas seções anteriores, existem vários fatores no sistema que contribuem com dificuldades no desenvolvimento da solução. Dessa forma, decidiu-se por tomar algumas limitações e hipóteses visando simplificar o sistema com o intuito de permitir testes preliminares e uma validação mais precisa do algoritmo desenvolvido.

Por isso, tomaram-se as seguintes limitações:

1. Os testes foram todos realizados em ambiente externo com o fim de possibilitar a máxima velocidade do obturador possível permitindo os 30 quadros por segundo;
2. As imagens foram capturadas por uma vista superior com o intuito de transformar o problema em um sistema 2D, como visto nas Figuras 23, 24 e 26, e pela facilidade em estimar o momento de impacto;
3. A câmera foi mantida estática com o objetivo de não ter outro movimento em cena além da bola;
4. A velocidade da bola foi limitada com o intuito de diminuir o erro causado pelo borrão detectado não prejudicando o funcionamento do sistema.

Figura 25 - Vista superior do ambiente que simula uma quadra de tênis e a linha da quadra.

Além disso, algumas hipóteses foram consideradas visando diminuir a complexidade do sistema:

1. Apesar de a bola deformar, ela foi considerada perfeitamente esférica no momento do impacto para checagem do contato ou não com a linha privando o sistema de ter que estimar a sua forma geométrica;
2. Mesmo com a vista superior, sabe-se que o diâmetro da bola em diferentes posições da quadra assume diferentes tamanhos. Entretanto, pelo fato dos testes terem sido feitos em um ambiente limitado, as variações são mínimas e o diâmetro foi considerado evitando de ter que estimar o diâmetro da bola a partir do borrão.

Com as condições de contorno e hipóteses estabelecidas, pode-se então partir para o desenvolvimento do algoritmo.

3.4. ALGORITMO

Apesar de o sistema ser de caráter generalizado e atender qualquer tipo de esporte com bola, é de se esperar que o algoritmo para cada caso tenha particularidades de tal modo a maximizar a qualidade do sistema. Além disso, por se tratar de um trabalho inicial, o projeto foi direcionado para um caso específico (bola de tênis) visando poder estudar a

teoria, analisar os métodos, desenvolver o *software* e interpretar os resultados esse trabalho de conclusão de curso.

O algoritmo foi desenvolvido de acordo com o fluxograma que pode ser visto na Figura 26.

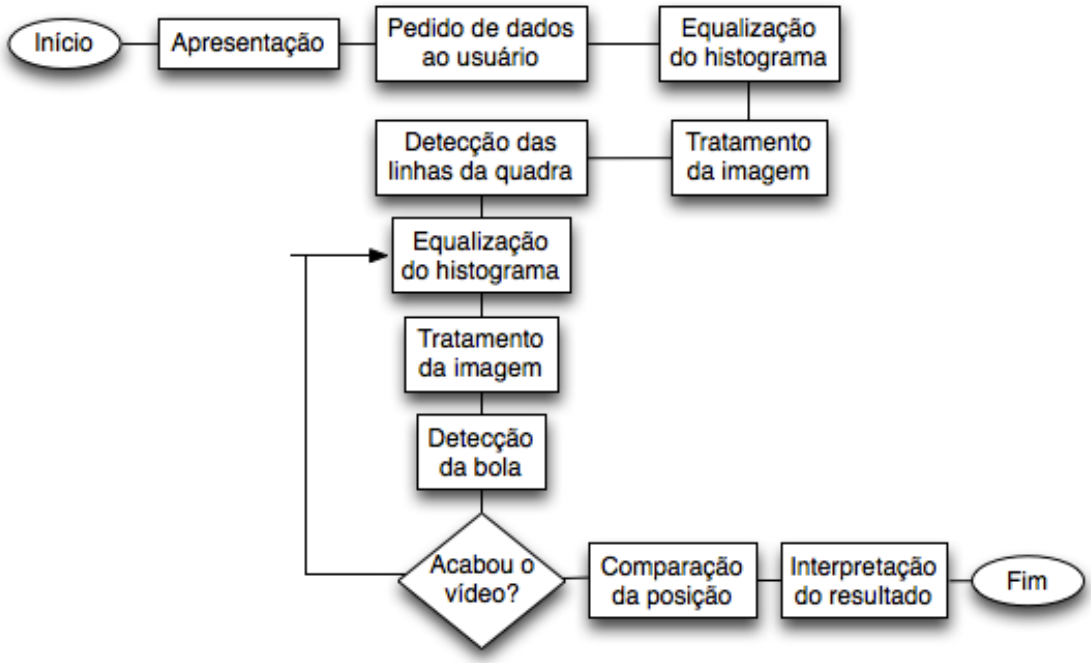

Figura 26 - Fluxograma do algoritmo desenvolvido.

Nas seções a seguir se dá o desenvolvimento do algoritmo.

3.4.1. INÍCIO DO PROGRAMA

O inicio do programa acontece com a apresentação usuário informando o título do trabalho, autor e outros dados importantes. O importante nessa parte é construir uma interface para que o programa possa ser usado por qualquer usuário sem experiência no assunto.

Buscando ser o mais geral possível, o programa pede ao usuário que entre com alguns dados como o nome do arquivo de vídeo a ser analisado. O programa analisará um vídeo por vez contendo uma situação em que a bola toca o chão próximo a linha da quadra. A imagem da interface em *MS-DOS* pode ser vista na Figura 27.

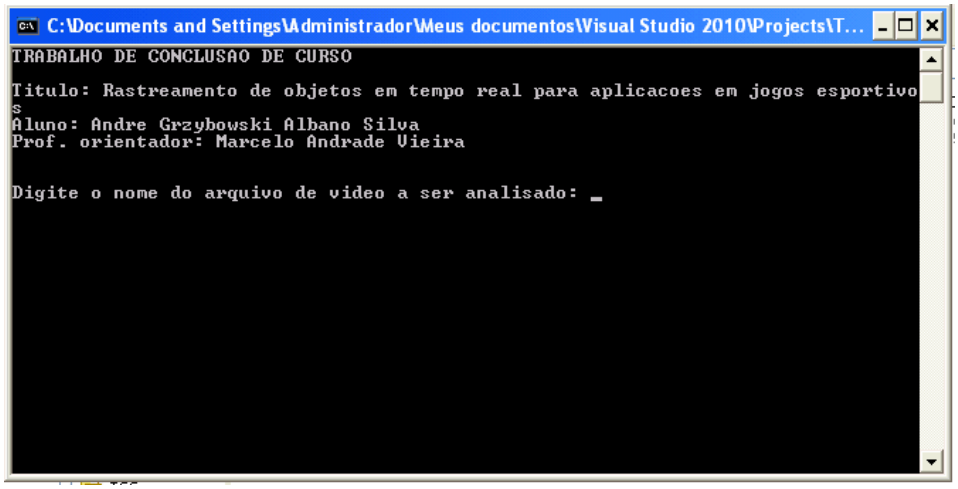

Figura 27 - Interface do programa.

3.4.2. ANÁLISE DO AMBIENTE

O algoritmo inicia analisando o ambiente. Como a câmera e as linhas estão estáticas se faz necessário a sua detecção apenas uma vez. Dessa forma, a captura de imagens se inicia instantes antes da bola entrar em movimento na quadra, assim, as linhas podem ser detectadas facilmente.

Inicialmente, é realizada a equalização do histograma da imagem. Essa operação visa compensar a falta ou excesso de luz na imagem. Felizmente, a biblioteca *OpenCV* possui uma função que permite realizar esse processo automaticamente, porém apenas um canal por vez. Como nesse caso queremos apenas detectar a posição da linha, pode ser utilizado qualquer canal RGB já que a linha é branca e a informação está presente em todos os canais. Nesse caso, foi escolhido o canal azul para a detecção das linhas e os canais vermelho e verde para a detecção da bola de tênis que é amarela.

A mesma operação de divisão de canais pode ser realizada utilizando o espaço de cores HSV. O canal de saturação exerce, no caso, uma função semelhante ao azul no RGB em que é simples realizar uma binarização do canal separando a bola da linha da quadra.

A seguir, as funções utilizadas nessa parte do algoritmo:

- cvSplit – Divide os canais em H, S e V ou R, G e B;
- cvCvtColor – Converte a imagem para um outro padrão de cor (preto e branco, RGB, HSV, etc);

- cvEqualizeHist – Realiza a equalização conforme a teoria.

Dessa forma, a divisão de canais pode ser facilmente realizada conforme pode ser visto o exemplo na Figura 28.

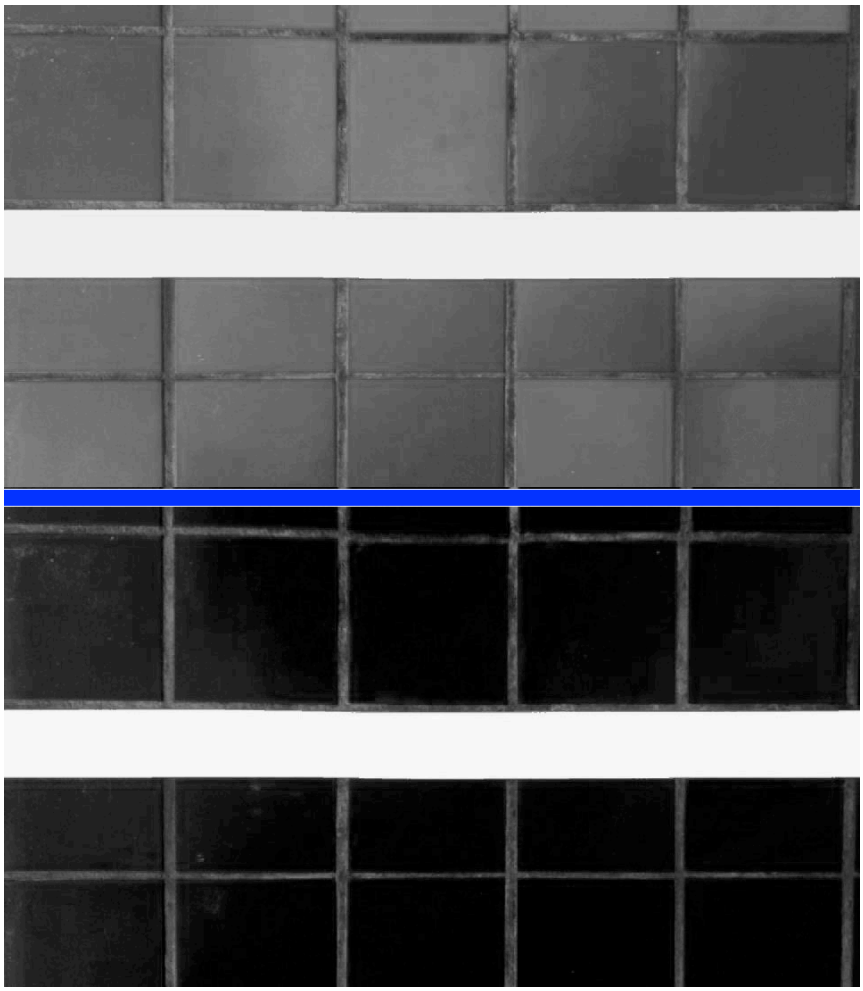

Figura 28 - Imagem dos canais vermelho (acima da linha azul) e azul (abaixo da linha azul).

Depois disso, o algoritmo irá detectar as linhas da quadra. O método utilizado foi binarização pelo fato de que se busca detectar uma região e por ser a única parte clara das imagens facilitando a segmentação e a escolha de um *threshold*. A imagem original e a binária podem ser vistas na Figura 29 abaixo.

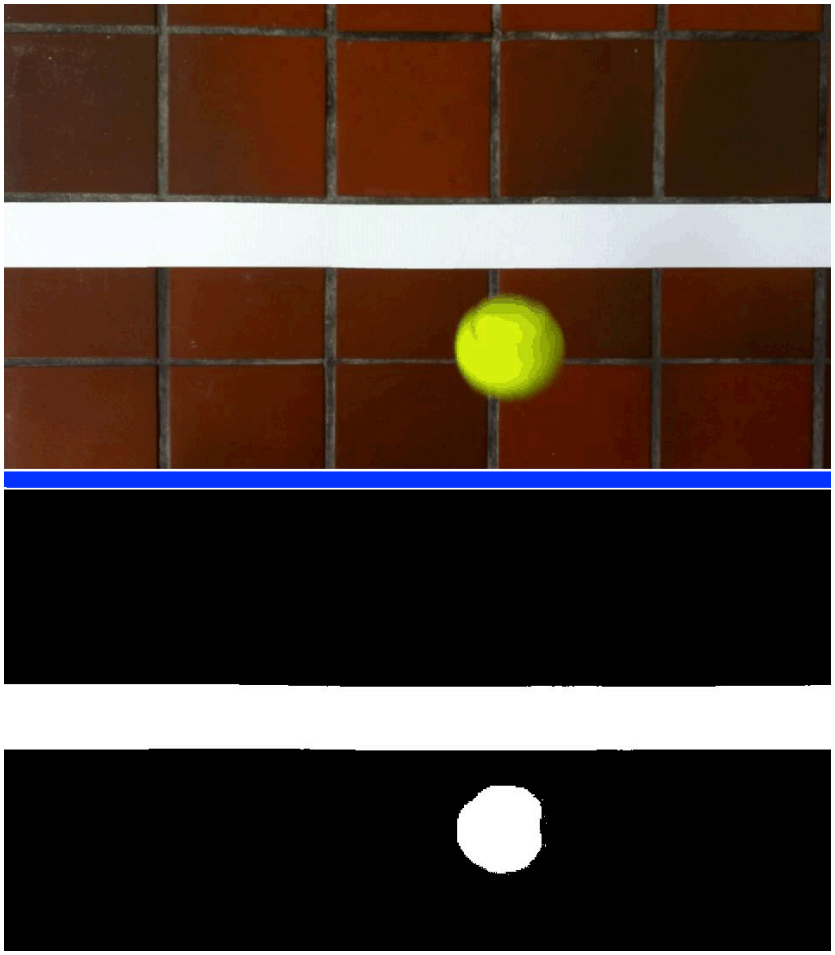

Figura 29 – Imagem da quadra (acima de linha azul) e da bola binarizada (abaixo da linha azul).

Após a binarização da imagem, realizou-se a operação morfológica de fechamento, para preencher os espaços vazios dentro da linha. Depois foi realizada uma operação de abertura linha buscando tirar o ruído no limite da linha com o resto da quadra. A razão do uso das operações após a binarização se deve ao fato que essas buscam otimizar a imagem binária. Além disso, pelo fato de as funções de operações morfológicas aceitarem apenas um canal por vez, é possível obter uma economia em processamento e memória.

Por fim, o algoritmo utiliza um filtro de suavização para eliminar os ruídos e aperfeiçoar a forma da bola e da linha facilitando, posteriormente, o rastreamento. Existe uma função que realiza a suavização por vários tipos chamada *cvSmooth*. Os filtros disponíveis são simples: simples sem escala, mediano, gaussiano e bilateral. Escolheu-se o gaussiano devido ao fato de ter melhores resultados em relação aos outros.

Um exemplo de suavização pode ser visto na Figura 30.

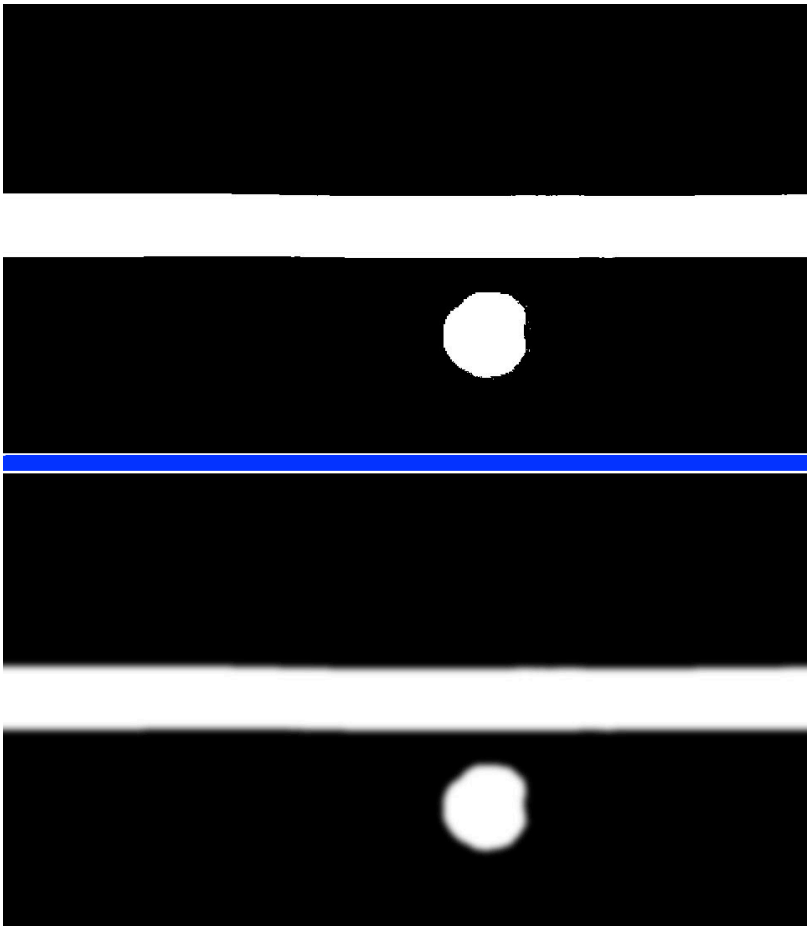

Figura 30 – Imagem binária (acima da linha azul) e suavizada (abaixo da linha azul).

A binarização foi feita com o *cvThreshold* e as operações morfológicas com *cvErosion*, *cvDilate* e *cvCreateStructuringElement*. Esse último cria o elemento estruturante que foi utilizado nas operações morfológicas. O uso de retângulos como elemento foi testado e otimizado seu tamanho para os resultados visualmente.

Para todas as operações acima (binarização, operações morfológicas e suavização), foi realizada uma inspeção da imagem de tal forma a poder determinar os parâmetros a serem utilizados nas funções e, após alguns testes, otimizar os seus valores. Entre os parâmetros estão nível de *threshold*, forma e tamanho do elemento estruturante e o filtro para suavização.

Assim, foi possível determinar a região pertencente às linhas da quadra no ambiente sendo considerado o ponto na vertical branco mais baixo o limite da linha da quadra.

3.4.3. ANÁLISE DA BOLA DE TÊNIS

Nessa parte, o algoritmo estará focado em continuamente checar pela segmentação da bola, o seu posicionamento quadro a quadro e realizar seu rastreamento. Assim, como no caso anterior, o algoritmo iniciará realizando a equalização do histograma e tratamento da imagem com as mesmas funções.

Posteriormente, partiu-se para a detecção da bola em si. Pelo fato da bola em alguns quadros aparecer em cima da linha na imagem, torna-se necessário que a linha seja excluída da imagem visando conseguir uma melhor imagem da bola e, assim, realizar sua detecção. Como é impossível realizar a binarização da imagem mantendo a bola e excluindo a linha, optou-se por realizar a binarização de canais diferentes da imagem colorida. O canal azul, que não contém a bola por ela ser amarela, detecta a linha apenas e o canal vermelho detecta a linha e a bola. Realizando a subtração do segundo canal pelo primeiro resulta em uma imagem contendo apenas a bola.. Para isso, usou-se a função *cvAbsDiff* que realiza a subtração entre duas imagens. Na Figura 31, pode ser visto o resultado dessa operação.

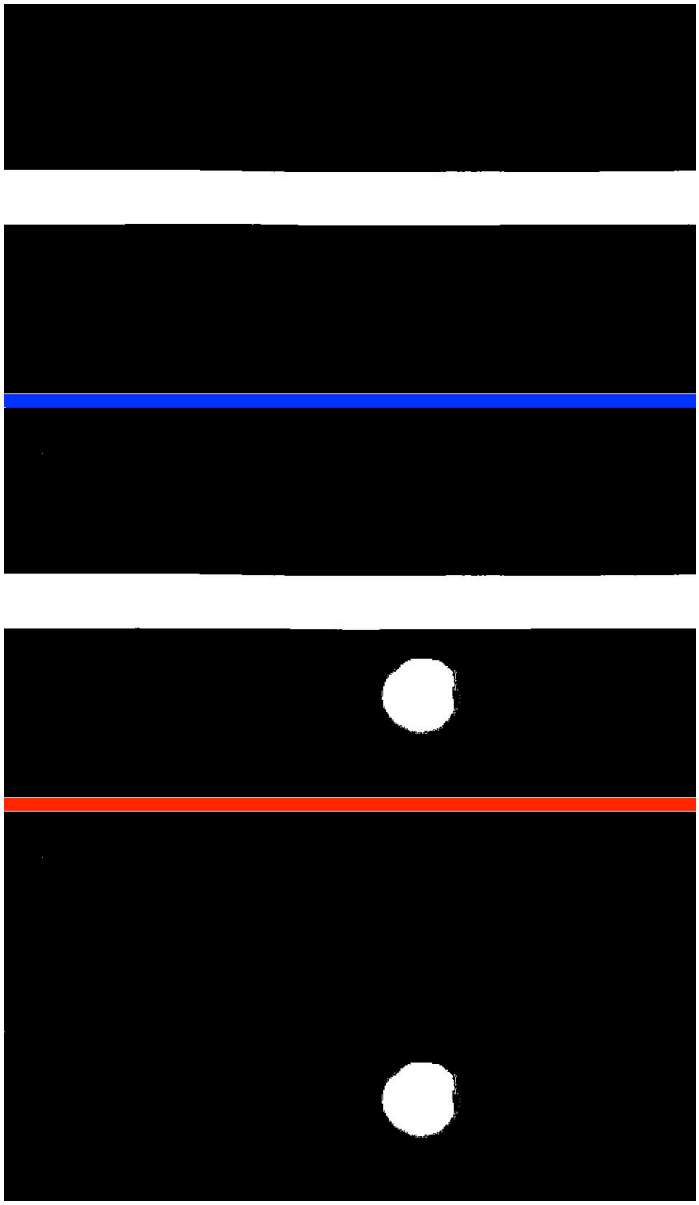

Figura 31 – Diferenciação dos canais sendo o canal azul (acima da linha azul), o canal vermelho (entre as linhas azul e vermelha) e a diferença dos canais (abaixo da linha vermelha) respectivamente.

Em alguns quadros, quando a movimentação da bola não está tão rápida e essa não está localizada acima da linha, é possível realizar a detecção por binarização, seguido de detector de bordas de Canny e transformada de Hough circular ou mesmo conexão de componentes. Porém, em casos que esses métodos falharem, deve-se utilizar algum método que modele a dinâmica da bola e preveja a localização no ponto em que não houve medida.

Os métodos testados no desenvolvimento do programa e que obtiveram resultados satisfatórios foram a transformada de Hough circular e a conexão de componentes. Para ajudar a modelar o sistema e prever o caminho da bola foi utilizado o filtro de Kalman.

3.4.3.1. TRANSFORMADA DE HOUGH

A transformada foi realizada com a função *cvHoughCircles,* de baixo custo computacional se comparados com transformada de Hough circulares comuns, que possui em sua definição a função de realizar o detector de bordas primeiro e com *threshold* apropriado detecta apenas a bola. Na Figura 32, abaixo, pode ser visto um exemplo de detecção do círculo referente à bola.

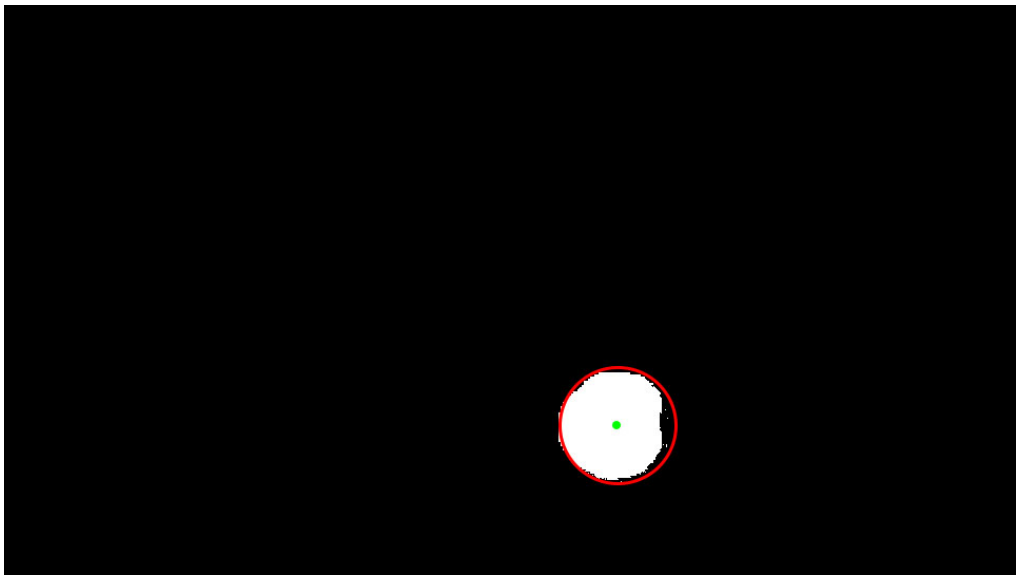

Figura 32 - Imagem da transformada de Hough com detector de Canny.

O método deve ser calibrado visando melhores resultados. Deve-se regular alguns parâmetros como o raio dos círculos a serem buscados (inspecionados em uma amostra de imagens capturadas), as distâncias entre os círculos encontrados em cada quadro (a maior possível, pois busca-se apenas um círculo por quadro) e a resolução do acumulador (ajustado para busca de círculos) usado para detectar os centros dos círculos. Essa calibração permite excluir falsos círculos e detectar com melhor precisão a bola na imagem.

3.4.3.2. CONEXÃO DE COMPONENTES

O método de conexão de componentes foi utilizado visando encontrar regiões, nesse caso, a região que representa a bola de tênis na imagem. Devido ao fato da imagem estar binarizada apenas com a bola e ter sido realizadas as operações de fechamento e aberturas, a região da bola fica bem definida e de fácil detecção pelo método.

Para aplicar o método foi utilizada a biblioteca *cvBlobsLib* que permite a utilização de funções que realizar a busca como é o caso da *CBlobResult.* Esse método, assim como a transformada de Hough, deve ser calibrado para atender as necessidades. Como ainda existem alguns ruídos na imagem, o parâmetro definido para filtragem foi a área referente a da região encontrada, o que permitiu detectar apenas a região da bola.

Na Figura 33, abaixo, é vista a detecção da região referente à bola.

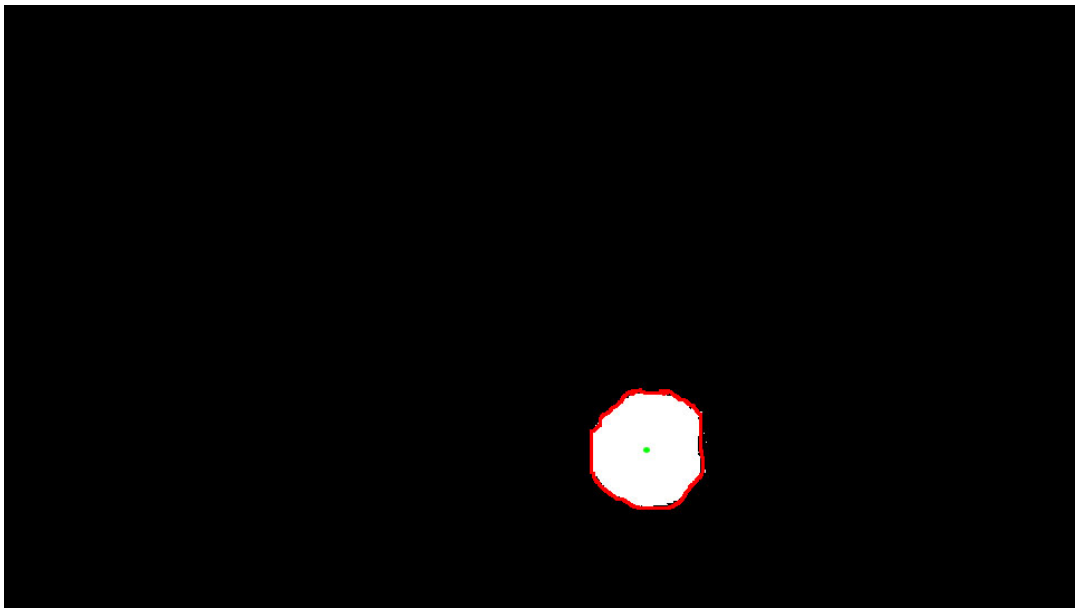

Figura 33 - Região encontrada por componentes conexos.

3.4.3.3. FILTRO DE KALMAN

Por fim, quando ocorre oclusão (a bola está em cima da linha e a subtração ocorre com muitos ruídos) e não é possível detectar a bola, utilizou-se o filtro de Kalman para estimar a posição da bola nessa situação.

O filtro de Kalman pode ser criado com a função *cvCreateKalman* em que se especifica a dimensão da variável de estado e da variável de medida. Dessa forma, a posição pode ser estimada com *cvKalmanPredict* e corrigida com as posições corretamente medida com *cvKalmanCorrect*. Se a estimativa e o valor medido divergirem muito, a medida teve muito ruído e é usado o valor da estimativa como posição da bola, senão será o valor medido. O filtro parte de um ponto inicial estimado através de análises humanas e a partir de medidas melhora o seu modelo.

Apesar de o filtro funcionar melhor com uma maior quantidade de amostras, esse se mostrou eficiente corrigindo rapidamente para um modelo muito próximo aos encontrados nas medidas tanto com a transformada de Hough como com conexão de componentes.

3.4.3.4. IMPACTO

Para a detecção do impacto utilizou-se o diâmetro da bola. Como a vista é superior, quanto mais próximo do chão a bola estiver, mais longe estará da câmera (que é fixa) e menor será seu diâmetro na imagem. Antes do algoritmo rodar, foi realizada uma detecção com bola parada e com a transformada de Hough e conexão de componentes foi determinado o diâmetro real da bola. Quando o diâmetro medido em movimento estiver em seu mínimo, ocorreu o impacto e a área usada para determinar se estava dentro ou fora será o centro da bola medido e o diâmetro determinado anteriormente.

Para interpretação dos resultados, no impacto a bola é calculada se está dentro ou fora, informado ao usuário o resultado, a bola é desenhada sobre a quadra e a imagem é salva para posterior conferência.

Na Figura 34, abaixo, um exemplo de resultado de rastreamento da bola por filtro de Kalman.

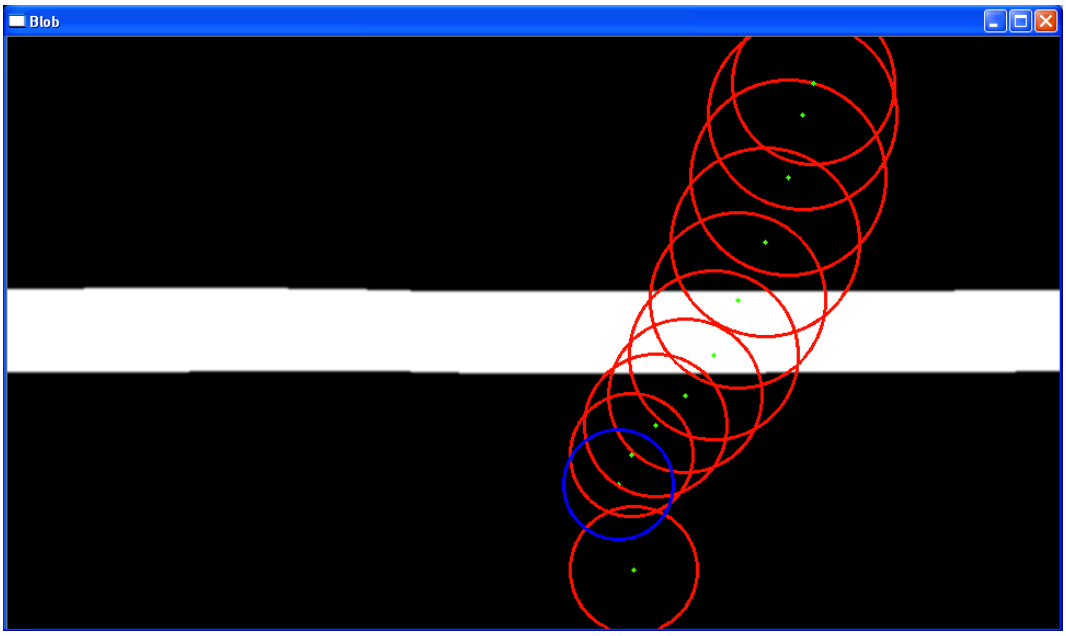

Figura 34 - Exemplo de ponto de impacto no círculo azul.

3.4.4. SAÍDAS DO ALGORITMO

O algoritmo fornece as informações de saída para o usuário da seguinte forma: na janelas *MS-DOS,* é passada aos usuários informações do ponto de impacto para cada método utilizado (transformada de Hough, conexão de componentes e filtro de Kalman) e se esta é considerada dentro ou fora de quadra de acordo com a linha.

Além disso, são criadas janelas, para todos os métodos, mostrando a imagem da quadra binarizada e desenhados todos os pontos nos quadros onde a bola foi detectada sendo marcado em azul o ponto considerado de impacto.

Por fim, é fornecida uma janela que permite ver o vídeo original quadro a quadro para realizar uma análise da eficiência de cada método verificando se o resultado foi correto ou não. Um exemplo da saída do algoritmo pode ser visto na Figura 35 e da saída em vídeo na Figura 36.

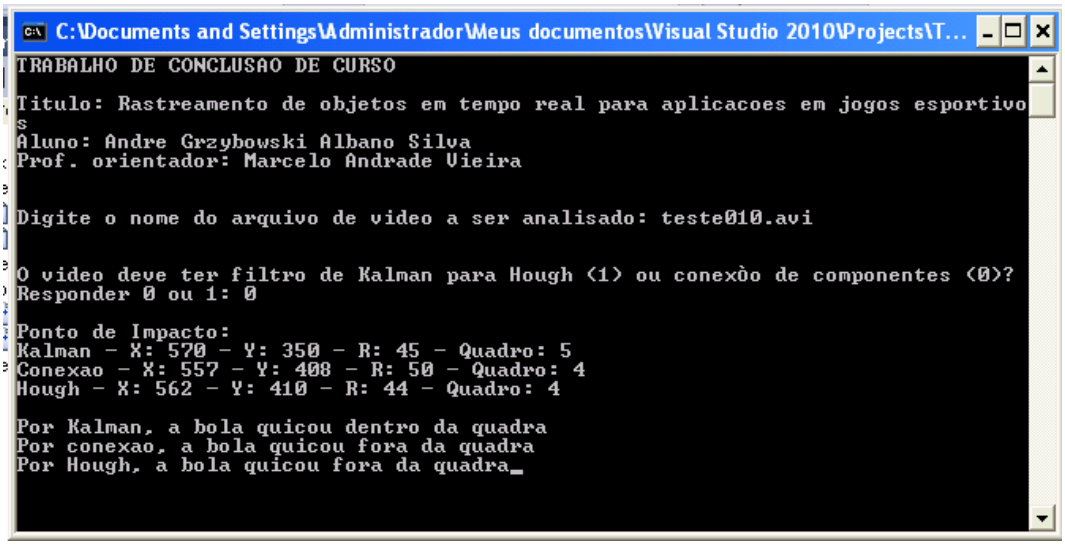

Figura 35 - Exemplo da saída do programa em DOS.

Figura 36 - Saída em vídeo com seleção de quadro.

3.4.5. CONSIDERAÇÕES FINAIS

Nessa seção foi abordada a solução proposta para o problema e como ela foi desenvolvida. O algoritmo completo pode ser visto no Apêndice A e na seção a seguir será feita uma análise dos resultados e dos métodos.

4. RESULTADOS E CONCLUSÕES

Para análise final da solução proposta, buscou-se utilizar um método que pudesse analisar a eficiência do algoritmo e o seu desempenho. Para isso foram realizados vários testes com o algoritmo buscando sempre variar a trajetória da bola para abranger o máximo de possibilidades possíveis.

Os resultados fornecidos pelo *software* foram analisados manualmente através da análise quadro a quadro dos vídeos e foi aferido visualmente se a detecção foi correta ou não. Posteriormente, foi feita uma análise com base na taxa de detecção do algoritmo.

A taxa de detecção (DR), comumente chamada de sensibilidade do sistema, foi calculada através da equação 24, onde TP vem de *true positive* e representa o número de casos em que a detecção foi realizada corretamente e FN significa *false negative* que representa quando a detecção falhou.

$$DR = \frac{TP}{TP+FN} \tag{24}$$

Testes foram realizados com 100 casos e, posteriormente, submetidos a essa análise. Buscou-se analisar todas as amostras no resultado que é de interesse para o projeto. Em outras palavras, se o método confirmou a bola como dentro ou fora e sua comparação com os resultados esperados.

Na Tabela 1, a seguir, pode ser visto os números obtidos com cada um dos métodos a partir da análise das amostras. O filtro de Kalman tem duas análises porque pode seguir medidas dos dois outros métodos.

	Acertos	Erros	Sensibilidade
Hough	90	10	90
Conexão	92	8	92
Kalman (Hough)	88	12	88
Kalman (conexão)	91	9	91

Tabela 1 - Tabela de erros e acertos dos métodos.

4.1 ANÁLISE DOS MÉTODOS

Nessa seção se realiza uma análise dos métodos utilizados, suas vantagens, desvantagens e eficiências.

4.1.1. TRANSFORMADA DE HOUGH

A transformada de Hough se mostrou uma ótima opção bastante viável no rastreamento da bola de tênis e seu trajeto. Quando bem calibrada, a transformada permite detectar o círculo que representa a bola em um nível bem próximo da realidade.

A grande vantagem desse método é sua capacidade de detectar bem próximo da perfeição o círculo quando há poucos ruídos devido a velocidade da bola ou interferências do ambiente. Entretanto, quando há a presença de ruídos, esse método se mostra desvantajoso por não entender algumas regiões como círculo ou entender como um de dimensões bem diferentes do real.

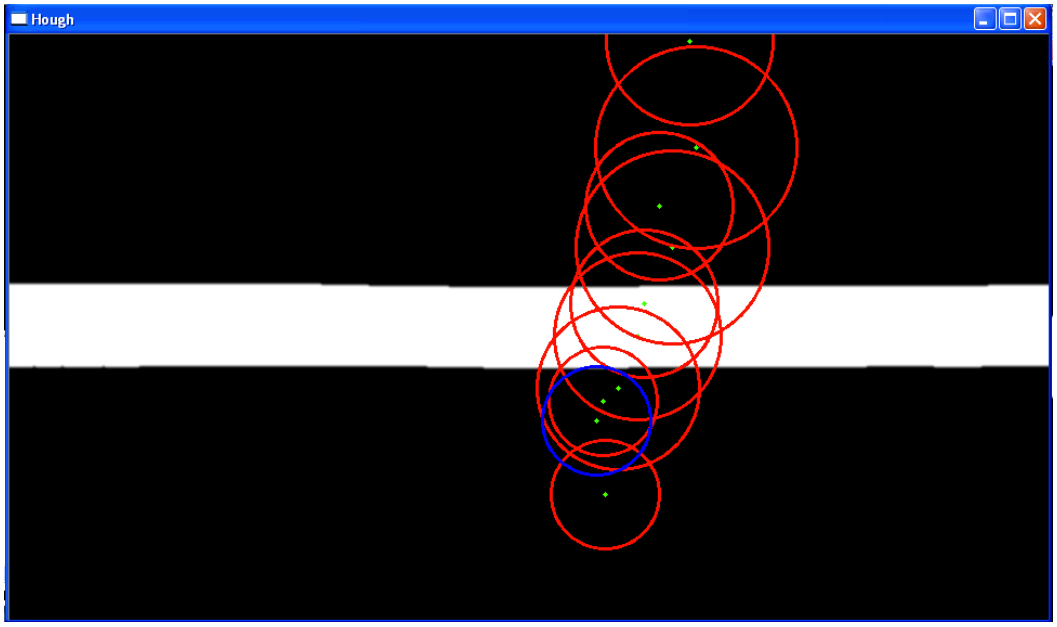

Figura 37 - Resultado de um vídeo pela transformada de Hough.

Na Figura 37, pode ser visto um resultado pela transformada de Hough de uma das filmagens. Analisando a distância em *pixels* de apenas o ponto de impacto de um dos testes, pode-se chegar ao erro da medidas dos círculos chegando aos resultados presentes na Tabela 2.

Real Diâmetro (pixels)	Hough Diâmetro (pixels)	Erro Pixels	Erro %
55	50	5	9,09

Tabela 2 - Tabela de erro da transformada de Hough.

Buscando analisar a eficiência do método, foi medida a sua velocidade de processamento. Para tal, o algoritmo foi modificado de tal forma a executar apenas a transformada de Hough sem o uso de um método auxiliar (com o tratamento das imagens e detecção do ponto de impacto) e se utilizou a função *clock()* e realizada a subtração entre o tempo após e anterior ao processamento. Para esse método, realizando essa medição com alguns testes o tempo de processamento médio foi de 1,015 segundos.

4.1.2. CONEXÃO DE COMPONENTES

O método de conexão de componentes se mostrou uma ótima opção no rastreamento da bola de tênis e seu trajeto. Excluindo os pequenos contornos de área pequena encontrados próximo a linha (sobras da subtração de imagens), as detecções são ótimas mesmo quando em cima da linha e com pouca variação do centro e raio do círculo em relação à realidade.

A grande vantagem desse método é sua capacidade de detectar a bola em praticamente todos os quadros. Sua capacidade de detectar qualquer região, mesmo que deformada, da bola é superior a transformada de Hough tornando esse método o com menos erros. Entretanto, por detectar regiões e não círculos propriamente ditos, seus resultados podem ser errados quando a região referente a bola está muito longe de um círculo tendo em vista que o raio é aproximado pela largura e comprimento da região e o centro é a centroide dessa.

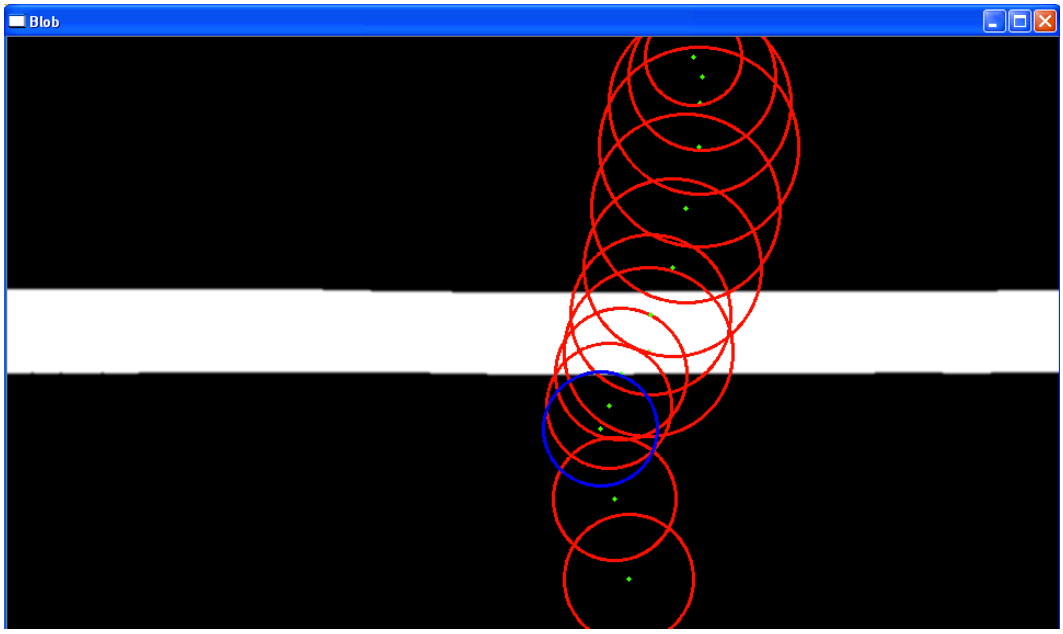

Figura 38 - Resultado por conexão de componentes.

Na Figura 38, pode ser visto um resultado por conexão de componentes de uma das filmagens. Analisando a distância em *pixels* de apenas o ponto de impacto de um dos testes, pode-se chegar ao erro da medidas dos círculos chegando aos resultados presentes na Tabela 3.

Real Diâmetro (pixels)	Conexão Diâmetro (pixels)	Erro Pixels	Erro %
55	52	3	5,45

Tabela 3 - Tabela de erro da conexão de componentes.

Buscando analisar a eficiência do método quanto ao tempo, foi medida a sua velocidade de processamento. Assim como no método anterior, o algoritmo foi modificado de tal forma a executar apenas a conexão de componentes sem o uso de um método auxiliar (com o tratamento das imagens e detecção do ponto de impacto) e se utilizou a função *clock()* e realizada a subtração entre o tempo após e anterior ao processamento. Para esse método, realizando essa medição com alguns testes o tempo de processamento médio foi de 0,828 segundos.

4.1.3. FILTRO DE KALMAN

O filtro de Kalman se mostrou uma opção no auxílio dos métodos acima no rastreamento da bola de tênis. Apesar do uso de poucos quadros, ele consegue se ajustar a

dinâmica da bola e realizar previsões com grande semelhança ao método de medida que ele está a seguir.

O filtro de Kalman, entretanto, depende do chute inicial para conseguir resultados satisfatórios ainda no começo da análise do vídeo. Apesar disso, ele consegue se adaptar bem ao método aplicado e frequentemente termina bem próximo ao método.

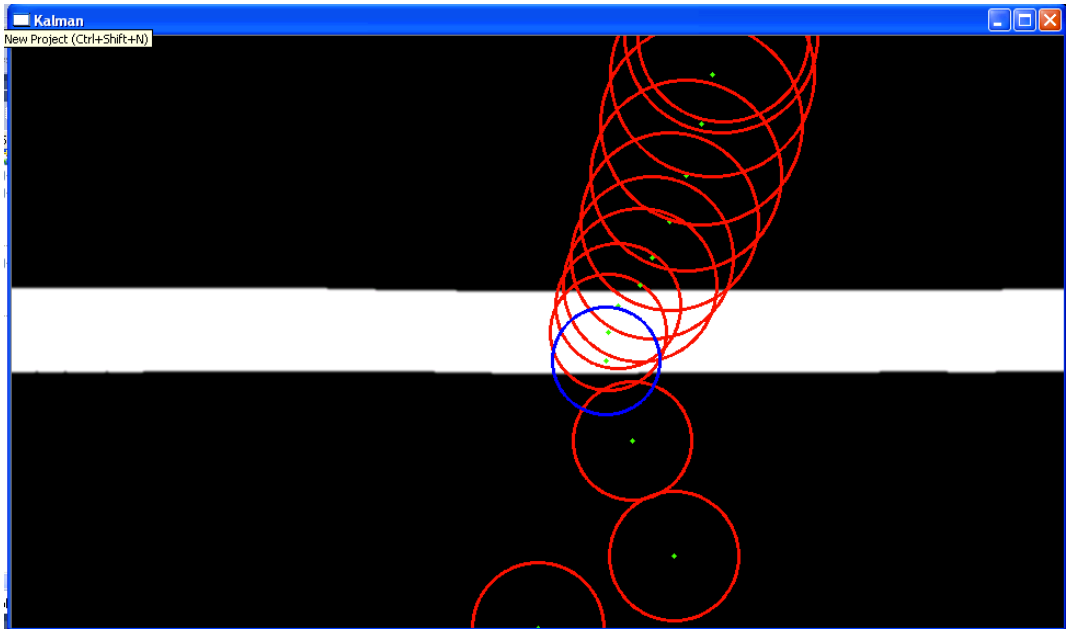

Figura 39 - Resultado por filtro de Kalman.

Na Figura 39, pode ser visto um resultado pelo filtro de Kalman de uma das filmagens. Analisando a distância em *pixels* de um dos círculos de apenas o ponto de impacto de um dos testes, pode-se chegar ao erro da medidas dos círculos chegando aos resultados presentes na Tabela 4.

Real Diâmetro (pixels)	Kalman Diâmetro (pixels)	Erro Pixels	Erro %
55	49	6	10,9

Tabela 4 - Tabela de erro do filtro de Kalman.

Assim como nos casos dos métodos anteriores, buscou-se analisar a eficiência do método quanto ao tempo. Para tal foi medida a sua velocidade de processamento de cada método (conexão de componentes e transformada de Hough) com a execução do filtro de Kalman. Utilizando a mesma função, foi possível calcular o tempo de processamento em alguns testes realizados. O valor médio encontrado para a execução em conjunto com a

transformada de Hough foi de 1,016 segundos e em conjunto com a conexão de componentes foi de 0,844 segundos. Esse resultado demonstra que o filtro de Kalman não interfere significativamente na redução da eficiência do algoritmo.

4.2. COMPARAÇÃO DOS MÉTODOS

Os métodos têm cada um suas vantagens e desvantagens e conforme, foi exemplificado nas figuras e tabelas, possuem erros aceitáveis dentro da proposta do projeto. Convertendo para o sistema métrico de medidas, os erros foram de 2,5 mm a 5,2 mm, o que pode ser considerado adequado, já que o sistema proposto apresenta uma série de limitações por simplicidade. Entretanto, essa análise vêm de apenas alguns exemplos e não de todas as amostras. Por isso, buscou-se analisar a eficiência do sistema com base no que é de interesse para o projeto como foi feito na Tabela 1.

Em relação aos resultados obtidos e precisão no rastreamento da bola, o método de conexão de componentes se mostrou a melhor opção. Além disso, o método se mostrou o mais rápido tendo um tempo de processamento menor sendo, então, uma ótima opção para aplicações em tempo real.

4.3. CONCLUSÃO

Analisando os resultados, pode-se concluir que o método proposto foi capaz de detectar satisfatoriamente a posição de impacto da bola com a linha de quadra. Testes com 100 vídeos mostram que o método proposto apresentou uma taxa de acerto de até 92%.

O pré-tratamento das imagens feitas quadro a quadro permitiu isolar e identificar as regiões pertencentes à linha e à bola para, posteriormente, realizar a análise com os métodos de rastreamento. A utilização da separação de canais, binarização, operações morfológicas, suavização e diferenciação de imagens se mostrou eficiente no tratamento das imagens visando permitir a detecção da linha e da bola.

Os métodos aplicados no rastreamento tiveram resultados satisfatórios. Apesar de algumas desvantagens de cada método, todos se mostraram eficientes conseguindo detectar a posição da bola com batante precisão se aproximando bastante da realidade. Além disso, todos se mostraram rápidos o suficiente para aplicações em tempo real.

Com relação aos outros projetos na área analisados, este trabalho conseguiu resultados em patamares semelhantes. Comparando com o projeto de Nibert e Spencer (2008), esse trabalho implementou um sistema mais avançado utilizando de mais métodos de rastreamento incluindo o filtro de Kalman como método auxiliar. Wu (2009) implementou um sistema semelhante a esse, porém fez uso de PCA para rastreamento da bola em virtude da utilização de uma câmera com maior taxa de amostragem que permitiu o uso do borrão da bola na sua detecção. Por fim, o projeto de Yan, Christmas e Kittler (2005) implementa um sistema mais avançado que consegue rastrear a bola em situações mais adversas (maior velocidade da bola e oclusões), porém trata-se de um trabalho mais complexo, desenvolvido com mais tempo.

Pode-se concluir que o sistema atende aos objetivos traçados no início deste trabalho sendo um sistema de visão computacional de baixo custo, utilizando filmadora e *hardware* de pequeno porte, capaz de detectar com precisão se a bola esteve em contato ou não com a linha da quadra. Como um trabalho inicial, foi possível detectar uma bola de tênis e seu impacto com a quadra em situações simplificadas podendo então ser expandido para situações mais generalizadas através de trabalhos futuros.

4.4. TRABALHOS FUTUROS

O sistema pode ser melhorado através do uso de uma câmera de melhor qualidade, principalmente quanto à velocidade de captura. Câmeras que pudessem capturar as imagens a uma taxa de 100 quadros por segundo pelo menos, com certeza melhorariam a detecção da bola possibilitando o uso do sistema em situações em que a bola estivesse com maior velocidade.

Outra sugestão seria a utilização de mais de uma câmera o que daria maior precisão no posicionamento da bola, permitiria fazer todo o rastreamento em 3D e melhoraria o problema de oclusão.

Visando maior precisão, poderia ser realizado o cálculo da deformação da bola no impacto permitindo calcular a área real tocada pela bola no momento do impacto, podendo dar um resultado mais completo se foi dentro ou fora da quadra.

Ainda buscando maior precisão do sistema, poderia ser levado em conta a trajetória da bola e a sua mudança de direção quadro a quadro de tal forma a auxiliar o algoritmo na

detecção do ponto de impacto. Em alguns testes, é comum ocorrer mudanças na direção do movimento da bola. Entretanto, isso não é uma regra. De tal forma, essa técnica serviria apenas como auxílio.

Uma outra forma de se desenvolver o sistema proposto seria a utilização da câmera próxima ao solo assim como no exemplo visto na Figura 1 no início deste trabalho. Esse posicionamento permitiria um rastreamento com maior precisão na detecção do ponto de impacto, mas traz maiores dificuldades no tratamento do ambiente ao fundo da imagem.

Por fim, o uso de inteligência artificial poderia deixar o sistema mais generalizado permitindo o seu uso em ambientes diferentes e até em esportes diferentes. A inteligência artificial poderia tornar os parâmetros das funções de binarização, tratamento da imagem (operações morfológicas e suavização) e técnicas de rastreamento (transformada de Hough, conexão de componentes e filtro de Kalman) variáveis podendo o sistema se adaptar a ambientes diferentes e até com bolas de tamanhos e cores diferentes.

5. REFERÊNCIAS

BRADSKI, G. ; KAEHLER, A. **Learning OpenCV Computer Vision with the OpenCV Library,** Sebastopol: O'Reilly Media, 2008.

CASTLEMAN, K. R. **Digital Image Processing,** New Jersey: Prentice Hall, Englewood Cliffs, 1996.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. New York: Addison-Wesley Publishing Company Inc, 2008.

Automated Line-Calling System: ITF Evaluation. Desenvolvido por ITF (*International Tennis Federation*), 2010. Disponível em <http://www.itftennis.com/shared/medialibrary/pdf/original/IO_5918_original.pdf>. Acesso em: 23 mar. 2011.

Line-calling systems – Introduction. Desenvolvido por ITF (*International Tennis Federation*), 2010. Disponível em <http://www.itftennis.com/technical/research/linecalling/>. Acesso em: 23 mar. 2011.

JACK, K. **Video Demystified: A Handbook for the Digital Engineer,** New South Wales: Newnes, 2007.

NIBERT, J. ; SPENCER, A. **Tennis Ball Tracker**. 2008. Rose-Hulman Institute of Technology. 2008.

PRATT, W. K. **Digital Image Processing,** New York: John Wiley & Sons Inc, 1991.

Introdução à Visão Computacional. Desenvolvido por VIEIRA, M. A. C., EESC/SEL, 2010. Disponível em <http://iris.sel.eesc.usp.br/sel339/>. Acesso em: 25 jan. 2011.

Visão Computacional. Desenvolvido por WANGENHEIM, A. V., UFSC, 2000. Disponível em <http://www.inf.ufsc.br/˜visão/2000/> em: 24 abr. 2011.

WELCH, G. ; BISHOP, G. **An Introduction to the Kalman Filter.** 2006. University of South Carolina at Chapel Hill. 2006.

WU, W. **Tennis Touching Point Detection based on High Speed Camera and Kalman Filter.** 2009. Dissertação (Mestrado) – Clemson University. 2009.

YAN, F. ; CHRISTMAS, W. ; KITTLER, J. **A Tennis Ball Tracking Algorithm for Automatic Annotation of Tennis Match.** 2005: Artigo – University of Surrey. 2005

APÊNDICE A – Código fonte do algoritmo

```
// TCC.cpp : Defines the entry point for the console application.
//

/* Trabalho de conclusão de curso

Aluno: André Grzybowski Albano Silva
No. USP: 5911113
Título: Rastreamento de objetos em tempo real para aplicações em jogos esportivos
Professor orientador: Marcelo Andrade da Costa Vieira */

#include "stdafx.h"
#include <math.h>

#include "highgui.h"   // Biblioteca para OpenCV
#include "cv.h"

#include "Blobresult.h"                  // Biblioteca para conexões de componentes

#include <iostream>            // Biblioteca para imprimir na tela
using namespace std;

// Declaração de variáveis globais

int slider = 0;
int flag2 = 0;
CvCapture* slideCapture = NULL;

// Declaração de estruturas

struct stateSpace {              // Estrutura de espaço de estado para as variáveis
        CvKalman* kalman;   // de dinâmica da bola de tênis
        CvMat* xk;
        CvMat* zk;
        CvMat* wk;
        const CvMat* yk;
};

struct image {
        IplImage* h;    // Imagem que guardará o resultado por Hough
        IplImage* b;    // Imagem que guardará o resultado por conexão de componentes
        IplImage* k;    // Imagem que guardará o resultado por Kalman
};

struct impact {          // Estrutura para armazenar ponto de impacto
        int x;
        int y;
        int d;
        int n;
};

// Declaração de funções
```

```
char* initProgram();    // Função de inicialização do programa e pede o nome do arquivo de vídeo
IplImage* frameTreatment(IplImage *src, int flag);  // Tratamento do quadro com equalização, binarização, operações morfológicas e suavização
IplImage* cvClose(IplImage* src);    // Operação fechamento
IplImage* cvOpen(IplImage* src);    // Operação abertura
int courtLimits(IplImage* src);                    // Define as regiões dentro e fora da quadra no quadro passado
struct impact impactPoint(float* p, struct impact s, int n);    // Determinação do ponto de impacto pela aceleração e diâmetro da bola
void outputProgram(struct image img, char* videoName, int nFrames, struct impact s[3], int limits); // Saídas do programa com resultados em DOS e visual
void onTrackbar(int pos);      // Função que controla a trackbar da saída do programa
struct stateSpace kalmanInit(int x00, int x01);         // Função inicializadora do filtro de Kalman
struct stateSpace kalmanCorrect(struct stateSpace s);      // Função que corrige as previsões do filtro de Kalman

int _tmain(int argc, _TCHAR* argv[]) {
        char* videoName;
        videoName = initProgram();  // Inicialização do programa pegando o nome do arquivo a ser analisado

        CvCapture* capture = cvCreateFileCapture(videoName);  // O vídeo é carregado

        int flag;          // Flag para determinar medidas de qual método o filtro de Kalman utilizará
        cout << "O video deve ter filtro de Kalman para Hough (1) ou conexão de componentes (0)? Responder 0 ou 1: ";
        cin >> flag;

        /*      O primeiro quadro de cada arquivo de vídeo não contém a bola
                ainda, apenas o ambiente. Dessa forma, esse é tratado e visto
                como apenas o fundo, ou seja, a quadra com a linha        */

        IplImage* courtTreated = cvQueryFrame(capture);
        courtTreated = frameTreatment(courtTreated,0);    // O primeiro quadro é tratado

        int limits = courtLimits(courtTreated);          // São calculados os limites da quadra

        // A imagem é convertida de volta ao espaço RGB visando a saída do programa
        struct image img;

        img.h = cvCreateImage(cvGetSize(courtTreated),IPL_DEPTH_8U,3);
        img.b = cvCreateImage(cvGetSize(courtTreated),IPL_DEPTH_8U,3);
        img.k = cvCreateImage(cvGetSize(courtTreated),IPL_DEPTH_8U,3);
        cvCvtColor(courtTreated,img.h,CV_GRAY2BGR);
        cvCvtColor(courtTreated,img.k,CV_GRAY2BGR);
        cvCvtColor(courtTreated,img.b,CV_GRAY2BGR);

        int nFrames = 1;        // Variável que guardará o número de quadros do vídeo para posterior uso

        struct impact impact[3];
```

```
        impact[0].d = 500;      // Variável que guardará os pontos de impacto
        impact[1].d = 500;      // Iniciado com raio alto para auxiliar a função
        impact[2].d = 500;      // no cálculo do ponto de impacto

        /*      Neste ponto, é inicializado o filtro de Kalman que será usado para auxiliar
                no rastreamento da bola. Cada variável tem seu filtro independente para
simplificar      */

        // Cada variável tem um estado de espaço de sua dinâmica
        struct stateSpace x = kalmanInit(480,0);      // Os parâmetros são inicializados
        struct stateSpace y = kalmanInit(540,-36);   // como ponto inicial, covariância
        struct stateSpace d = kalmanInit(60,-5);              // e outros

        /*      Neste ponto, inicia-se o loop que irá analisar os quadros restantes em busca
                da bola de tênis. Os quadros serão tratados e analisados para encontrar a
bola
                em cada quadro, determinar o seu ponto de impacto e checar se está dentro
ou fora.        */

        while(1) {
                IplImage* frameTreated = cvQueryFrame(capture);         // Captura o
próximo quadro
                if(!frameTreated) break;                                        // e checa
sua existência

                /* 1.    FILTRO DE KALMAN: O primeiro método de rastreamento
                                é na verdade um método para auxiliar os outros métodos
                                em casos que ocorrerem falhas ou oclusão nas medidas
                                prevendo a localização da bola com base em medidas
anteriores.

                                Ele pode ser usado para um dos dois métodos a ser escolhido
no
                                inicio do programa.    */

                x.yk = cvKalmanPredict(x.kalman,0);         // A previsão é feita para cada
variável;
                y.yk = cvKalmanPredict(y.kalman,0);
                d.yk = cvKalmanPredict(d.kalman,0);

                //cout << "x: " << x.yk->data.i[0] << " y: " << y.yk->data.i[0] << " d: " << d.yk-
>data.i[0] << endl;
                float pfl[3] = {(float)x.yk->data.i[0],(float)y.yk->data.i[0],(float)d.yk->data.i[0]};
                CvPoint center = cvPoint(cvRound(pfl[0]),cvRound(pfl[1]));          // O ponto
previsto é desenhado
                cvCircle(img.k,center,2,CV_RGB(0,255,0),-1);
        // na respectiva imagem
                cvCircle(img.k,center,cvRound(pfl[2]),CV_RGB(255,0,0),2);

                impact[0] = impactPoint(pfl,impact[0],nFrames);             // Chamada função
para cálculo do ponto de impacto



                /*      Nesse ponto o quadro é tratado de forma que fique simples para os
```

```
                        métodos de Hough e contornos consigam buscar a bola de tênis.
                        Como a bola é amarela e a linha é branca, obtém-se uma imagem
binarizada
                        do canal vermelho da imagem em que aparecerá a bola e a linha em
branco
                        (devido ao nível de threshold) e uma imagem binarizada do canal azul
em
                        que só aparecerá a linha (o amarelo está fora do canal azul).
                        Dessa forma, pode ser feita uma subtração das imagens para termos
apenas
                        a bola na imagem final. Isso evita interferências quando a bola está
em
                        cima da linha na imagem original.    */

                courtTreated = frameTreatment(frameTreated,0);
                frameTreated = frameTreatment(frameTreated,1);  // Obtenção do quadro
tratado

                IplImage* frameDiff =
cvCreateImage(cvGetSize(frameTreated),IPL_DEPTH_8U,1);
                cvAbsDiff(courtTreated,frameTreated,frameDiff);    // Subtração das duas
imagens

                /*      2.      CONEXÃO DE COMPONENTES: Esse método utiliza a
biblioteca de OpenCV, cvBlobsLib,
                        que, através da conexão de componentes encontra os borrões
presentes na imagem.
                        Com uso de um filtro por área, é possível encontrar apenas o
borrão correspondente
                        a bola de tênis.        */

                CBlobResult blobs;    // Inicialização das variáveis
                CBlob *currentBlob;

                blobs = CBlobResult(frameDiff,NULL,0);     // Busca pela bola na imagem

                blobs.Filter(blobs,B_EXCLUDE,CBlobGetArea(),B_LESS,4000);  // Filtro por
área para apenas encontrar a bola

                if(blobs.GetNumBlobs() > 0) {         // Apenas se encontrou algum indicio da
bola
                        currentBlob = blobs.GetBlob(0);       // Apenas a resposta principal é
considerada (mais próxima da bola)

                        float p[3] = {currentBlob->GetEllipse().center.x,currentBlob-
>GetEllipse().center.y,(currentBlob->GetEllipse().size.height+currentBlob-
>GetEllipse().size.width)/2};
                        CvPoint center = cvPoint(cvRound(p[0]),cvRound(p[1]));              // O
ponto encontrado
                        cvCircle(img.b,center,2,CV_RGB(0,255,0),-1);
                                                                                            // é
desenhado
                        cvCircle(img.b,center,cvRound(p[2]),CV_RGB(255,0,0),2);
```

```
                        impact[1] = impactPoint(p,impact[1],nFrames);


                        if(flag == 0) {
                                x.zk->data.i[0] = cvRound(p[0]);                // Correção do filtro
de Kalman
                                y.zk->data.i[0] = cvRound(p[1]);
                                d.zk->data.i[0] = cvRound(p[2]);

                                x = kalmanCorrect(x);
                                y = kalmanCorrect(y);
                                d = kalmanCorrect(d);
                        }
                }

                /*      3.      TRANSFORMA DE HOUGH: A transformada de Hough para
círculos é realizada buscando as bolas
                                de tênis na imagem. A função utilizada da biblioteca OpenCV
já realiza o detector de bordas
                                de Canny na imagem. As possíveis bolas são filtradas pelo raio
e outros parâmetros do método
                                evitando falsos círculos */

                CvMemStorage* storage = cvCreateMemStorage(0);        // Método aplicado
                CvSeq *circles =
cvHoughCircles(frameDiff,storage,CV_HOUGH_GRADIENT,2,cvGetSize(frameDiff).width/1
0,100,50,40,120);

                if(circles->total > 0) {            // Apenas se houver círculos
                        float* p = (float*) cvGetSeqElem(circles,0);                    // Apenas a
bola principal é considerada, fugindo de falsos círculos
                        CvPoint center = cvPoint(cvRound(p[0]),cvRound(p[1]));    // O ponto e
diâmetro é marcado na imagem para análise manual na saída depois
                        cvCircle(img.h,center,2,CV_RGB(0,255,0),-1);
                        cvCircle(img.h,center,cvRound(p[2]),CV_RGB(255,0,0),2);

                        impact[2] = impactPoint(p,impact[2],nFrames);


                        if(flag == 1) {
                                x.zk->data.i[0] = cvRound(p[0]);
                                y.zk->data.i[0] = cvRound(p[1]);
                                d.zk->data.i[0] = cvRound(p[2]);

                                x = kalmanCorrect(x);
                                y = kalmanCorrect(y);
                                d = kalmanCorrect(d);
                        }
                }

                nFrames++;    // Aumenta a contagem de quadros do vídeo
        }

        outputProgram(img,videoName,nFrames,impact,limits);    // Dá as saídas do
```

```
programa

	cvWaitKey();  // Mantém o programa aberto
	return 0;
}

char* initProgram() {
	char* videoName = (char*)malloc(20*sizeof(char)); // Variável que guardará o nome
do arquivo

	cout << "TRABALHO DE CONCLUSAO DE CURSO\n\n";
	cout << "Titulo: Rastreamento de objetos em tempo real para aplicacoes em jogos
esportivos\n";
	cout << "Aluno: Andre Grzybowski Albano Silva\n";
	cout << "Prof. orientador: Marcelo Andrade Vieira\n\n\n";
	cout << "Digite o nome do arquivo de video a ser analisado: ";

	cin >> videoName;

	cout << "\n\n";

	return videoName;
}

IplImage* frameTreatment(IplImage* src, int flag) {
	IplImage* out = cvCreateImage(cvGetSize(src),IPL_DEPTH_8U,1);

	// Divisão da imagem no seus 3 canais RGB
	if(flag) cvSplit(src,out,NULL,NULL,NULL);   // Vermelho para a bola
	else cvSplit(src,NULL,NULL,out,NULL);       // Azul para o fundo da quadra
	cvEqualizeHist(aux,out);

	cvThreshold(out,out,128,255,CV_THRESH_BINARY);      // Binarização da imagem

	out = cvClose(out);    // Operação fechamento
	out = cvOpen(out);    // Operação abertura

	cvSmooth(out,out,CV_GAUSSIAN,5,5);      // Suavização da imagem

	return out;
}

IplImage* cvClose(IplImage* src) {
	IplConvKernel *se =
cvCreateStructuringElementEx(21,21,10,10,CV_SHAPE_RECT,NULL);

	cvErode(src,src,se,1);                  // O fechamento é composto da erosão
	cvDilate(src,src,se,1);          // seguido da dilatação

	cvReleaseStructuringElement(&se);
	return src;
}

IplImage* cvOpen(IplImage* src) {
	IplConvKernel *se =
```

```
cvCreateStructuringElementEx(11,11,5,5,CV_SHAPE_RECT,NULL);

	cvErode(src,src,se,1);		// A abertura já é o contrário do fechamento
	cvDilate(src,src,se,1);

	cvReleaseStructuringElement(&se);
	return src;
}

int courtLimits(IplImage* src) {
	int limits = 0;

	/*	O algoritmo busca pelo ponto branco mais baixo na imagem,
		ou seja, o ponto da linha que determina o limite da quadra.
		Esse y será considerado o limite.	*/

	for(int x = 0; x < cvGetSize(src).width; x++) {
		for(int y = 0; y < cvGetSize(src).height; y++) {
			if(cvGet2D(src,y,x).val[0] > 0) {
				limits = y;
			}
		}
	}

	return limits;
}

void onTrackbar(int pos) {
	cvSetCaptureProperty(slideCapture,CV_CAP_PROP_POS_FRAMES,pos);
	IplImage* frame = cvQueryFrame(slideCapture);	// Trackbar que mostra os
quadros da imagem

	// permitindo ver a imagem quadro a quadro

	// com calma para análise
	cvShowImage("Video",frame);
}

void outputProgram(struct image img, char* videoName, int nFrames, struct impact s[3], int
limits) {
	// Saída na janela DOS a respeito do ponto de impacto
	cout << "\nPonto de Impacto:\n";
	cout << "Kalman - X: " << s[0].x << " - Y: " << s[0].y << " - R: " << s[0].d << " -
Quadro: " << s[0].n << endl;
	cout << "Conexao - X: " << s[1].x << " - Y: " << s[1].y << " - R: " << s[1].d << " -
Quadro: " << s[1].n << endl;
	cout << "Hough - X: " << s[2].x << " - Y: " << s[2].y << " - R: " << s[2].d << " - Quadro:
" << s[2].n << endl;

	if(s[0].y-s[0].d > limits) cout << "\nPor Kalman, a bola quicou fora da quadra";
	else cout << "\nPor Kalman, a bola quicou dentro da quadra";

	if(s[1].y-s[1].d > limits) cout << "\nPor conexao, a bola quicou fora da quadra";
	else cout << "\nPor conexao, a bola quicou dentro da quadra";
```

```
        if(s[2].y-s[2].d > limits) cout << "\nPor Hough, a bola quicou fora da quadra";
        else cout << "\nPor Hough, a bola quicou dentro da quadra";

        // Marcação do ponto de impacto nas imagens
        CvPoint center = cvPoint(cvRound(s[0].x),cvRound(s[0].y));
        cvCircle(img.k,center,cvRound(s[0].d),CV_RGB(0,0,255),2);

        center = cvPoint(cvRound(s[1].x),cvRound(s[1].y));
        cvCircle(img.b,center,cvRound(s[1].d),CV_RGB(0,0,255),2);

        center = cvPoint(cvRound(s[2].x),cvRound(s[2].y));
        cvCircle(img.h,center,cvRound(s[2].d),CV_RGB(0,0,255),2);

        cvNamedWindow("Hough",CV_WINDOW_AUTOSIZE);
        cvShowImage("Hough",img.h);

        cvNamedWindow("Kalman",CV_WINDOW_AUTOSIZE);
        cvShowImage("Kalman",img.k);

        cvNamedWindow("Blob",CV_WINDOW_AUTOSIZE);
        cvShowImage("Blob",img.b);

        // Ativação do vídeo com trackbar para análise dos resultados
        cvNamedWindow("Video",CV_WINDOW_AUTOSIZE);
        slideCapture = cvCreateFileCapture(videoName);
        cvCreateTrackbar("Posicao","Video",&slider,nFrames,onTrackbar);
        onTrackbar(0);
}

struct impact impactPoint(float* p, struct impact s, int n) {
        /*      Caso já haja algumm ponto de mínimo, outro ponto de mínimo
                será descartado, sendo considerado um erro já que a bola está
                subindo de novo.        */

        if(p[2] > s.d) flag2 = 1;
        else if((p[2] < s.d) && (flag2 == 0)) { // Caso contrário, o menor ponto de raio será
                s.x = cvRound(p[0]);                            // considerado ponto de impacto.
                s.y = cvRound(p[1]);
                s.d = cvRound(p[2]);
                s.n = n;
        }

        return s;
}

struct stateSpace kalmanInit(int x00, int x01) {
        struct stateSpace s;

        // Inicialização de Kalman para uma das variáveis
        s.kalman = cvCreateKalman(2,1,0);

        // Matriz de transição (matriz A das equações)
        const float F[] = {1,1,0,1};
        memcpy(s.kalman->transition_matrix->data.fl,F,sizeof(F));
```

```
        // Matrizes H, erro e ruídos
        cvSetIdentity(s.kalman->measurement_matrix,cvRealScalar(1));
        cvSetIdentity(s.kalman->process_noise_cov,cvRealScalar(1e-5));
        cvSetIdentity(s.kalman->measurement_noise_cov,cvRealScalar(1e-1));
        cvSetIdentity(s.kalman->error_cov_post,cvRealScalar(1));

        // Inicialização das matrizes de previsão, medição e correção
        s.xk = cvCreateMat(2,1,CV_32FC1);
        s.wk = cvCreateMat(2,1,CV_32FC1);
        s.zk = cvCreateMat(1,1,CV_32FC1);

        // Dados iniciais mais comuns esperados
        s.xk->data.i[0] = x00;
        s.xk->data.i[1] = x01;
        memcpy(s.kalman->state_post->data.i,s.xk->data.i,sizeof(s.xk->data.i));

        s.wk->data.i[0] = cvRound(0.01);
        cvZero(s.zk);

        return s;
}

struct stateSpace kalmanCorrect(struct stateSpace s) {
        cvKalmanCorrect(s.kalman,s.zk);              // Correção de Kalman
        cvMatMulAdd(s.kalman->transition_matrix,s.xk,s.wk,s.xk);          // Correção do valor
xk

        return s;
}
```