UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

GUSTAVO ILHA

RASTREAMENTO AUTOMÁTICO DA BOLA DE FUTEBOL EM VÍDEOS

Porto Alegre
2009

GUSTAVO ILHA

RASTREAMENTO AUTOMÁTICO DA BOLA DE FUTEBOL EM VÍDEOS

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Área de concentração: Tecnologia de Informação e Comunicações

ORIENTADOR: Altamiro Amadeu Susin

Porto Alegre

2009

GUSTAVO ILHA

RASTREAMENTO AUTOMÁTICO DE OBJETOS EM VÍDEOS:
APLICAÇÃO AO RASTREAMENTO DA BOLA EM JOGOS DE FUTEBOL

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Altamiro Amadeu Susin, UFRGS

Doutor pelo Instituto Nacional Politécnico de Grenoble – Grenoble, França

Banca Examinadora:

Prof. Dr. Marcelo Lubaszewski, PPGEE – UFRGS

Doutor pela Ecoles d'Ingénieurs et Formations de Docteurs -Grenoble, França

Prof. Dr. Sérgio Bampi, PPGC – UFRGS

Doutor pela Stanford University – Palo Alto, Estados Unidos

Prof. Dra. Letícia Guimarães, LAPSI – UFRGS

Doutor pela Muroran Institute of Technology – Muroran, Japão

Coordenador do PPGEE:

_____

Prof. Dr. Arturo Suman Bretas

Porto Alegre, 12 de Janeiro de 2009

Dedico este trabalho aos meus pais, em especial pela dedicação e apoio em todos os momentos difíceis.

AGRADECIMENTOS

Ao Programa de Pós-Graduação em Engenharia Elétrica, PPGEE, pela oportunidade de realização do trabalho de pesquisa na área de Processamento de Imagens.

Aos colegas do LAPSI pelo seu auxílio nas tarefas desenvolvidas durante o curso e apoio na revisão deste trabalho.

À FIFA® pela gentileza de ceder um vídeo de seu acervo para a realização dos testes do algoritmo.

À Manuela, pelo apoio em todos os momentos.

RESUMO

A localização de objetos em uma imagem e acompanhamento de seu deslocamento numa sequência de imagens são tarefas de interesse teórico e prático. Aplicações de reconhecimento e rastreamento de padrões e objetos tem se difundido ultimamente, principalmente no ramo de controle, automação e vigilância. Esta dissertação apresenta um método eficaz para localizar e rastrear automaticamente objetos em vídeos. Para tanto, foi utilizado o caso do rastreamento da bola em vídeos esportivos, especificamente o jogo de futebol. O algoritmo primeiramente localiza a bola utilizando segmentação, eliminação e ponderação de candidatos, seguido do algoritmo de Viterbi, que decide qual desses candidatos representa efetivamente a bola. Depois de encontrada, a bola é rastreada utilizando o Filtro de Partículas auxiliado pelo método de semelhança de histogramas. Não é necessária inicialização da bola ou intervenção humana durante o algoritmo. Por fim, é feita uma comparação do Filtro de Kalman com o Filtro de Partículas no escopo do rastreamento da bola em vídeos de futebol. E, adicionalmente, é feita a comparação entre as funções de semelhança para serem utilizadas no Filtro de Partículas para o rastreamento da bola. Dificuldades, como a presença de ruído e de oclusão, tanto parcial como total, tiveram de ser contornadas.

Palavras-chaves: Processamento de Imagens. Processamento de Sinais. Filtro de Kalman. Filtro de Partículas. Rastreamento de objetos. Algoritmo de Viterbi.

ABSTRACT

The location of objects in an image and tracking its movement in a sequence of images is a task of theoretical and practical interest. Applications for recognition and tracking of patterns and objects have been spread lately, especially in the field of control, automation and vigilance. This dissertation presents an effective method to automatically locate and track objects in videos. Thereto, we used the case of tracking the ball in sports videos, specifically the game of football. The algorithm first locates the ball using segmentation, elimination and the weighting of candidates, followed by a Viterbi algorithm, which decides which of these candidates is actually the ball. Once found, the ball is tracked using the Particle Filter aided by the method of similarity of histograms. It is not necessary to initialize the ball or any human intervention during the algorithm. Next, a comparison of the Kalman Filter to Particle Filter in the scope of tracking the ball in soccer videos is made. And in addition, a comparison is made between the functions of similarity to be used in the Particle Filter for tracking the ball. Difficulties, such as the presence of noise and occlusion, in part or in total, had to be circumvented.

Keywords: Electrical Engineering. Signal Processing. Image Processing. Kalman Filter. Particle Filter. Object Tracking. Viterbi Decoder.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API – Interface de Programação de Aplicativos, do inglês Application Programming Interface

HSI – Espaço de cores: Matiz, Saturação e Intensidade, do inglês Hue, Saturation and Intensity

LAPSI – Laboratório de Processamento de Sinais e Imagens (UFRGS)

LILI2 – Biblioteca de Processamento de Imagens do Lapsi em linguagem C++

MBR – Retângulo Mínimo Circunscrito, do inglês Minimal Bounding Rectangle

RGB – Vermelho, Verde e Azul, do inglês Red, Green and Blue

SDK – Kit de Desenvolvimento de Software, do inglês Software Development Kit

WXY – Espaço de Cores desenvolvido no LAPSI

YCbCr – Espaço de cores: Luminância, Complemento de Azul e Complemento de Vermelho

SUMÁRIO

1 INTRODUÇÃO

Com a popularização de equipamentos capazes de gerar, armazenar e transmitir vídeos, o tema de processamento de imagens possui um novo papel na sociedade. A utilização de métodos para obter informações automaticamente de vídeos e de seqüência de imagens está se tornando cada vez mais comum.

A grande quantidade de dados aliada com a rapidez exigida para que as novas informações sejam geradas e distribuídas abre caminho para o processamento automático desses dados. As informações extraídas das seqüências de vídeos são utilizadas das mais variadas formas: geração de estatísticas, classificação, reconhecimento de pessoas ou objetos são alguns exemplos de aplicações apresentadas pelos algoritmos.

A utilização de algoritmos de rastreamento de objetos é presente numa gama muito ampla de setores da atividade humana. Em sistemas bélicos é de fundamental importância o correto rastreamento de possíveis alvos. Na indústria, utilizam-se tais sistemas para verificação da qualidade de produtos bem como para acompanhamento dos mesmos na linha de produção. No ramo de segurança, a detecção e rastreamento de pessoas podem ajudar a identificar a presença de indivíduos não autorizados.

O processamento de vídeos de eventos esportivos tem se tornado item de interesse global nas duas últimas décadas, automatizando o processo que normalmente era feito manualmente após o fim do jogo. Isso permite que as informações pertinentes sejam geradas ao vivo e muitas vezes em tempo real. O futebol, por ser um dos esportes mais populares do mundo, sempre obteve lugar de destaque nas pesquisas do tema.

O principal benefício do processamento de vídeos esportivos é a obtenção automática de informações sobre o evento. A localização da bola, a localização dos jogadores, a localização do campo, são bons exemplos de informações preciosas que podem ser obtidos de tais vídeos. Com esses dados disponíveis, podem-se gerar diversas aplicações, como por exemplo, análises táticas automatizadas e em tempo real, controle automático das câmeras, seleção automática de melhores momentos das partidas, geração automática de pan-scan para TV digital, etc.

A comunidade científica, ciente desse processo, vem publicando diversos artigos na área. Diversos algoritmos foram desenvolvidos para resolver os problemas até aqui encontrados.

O problema de rastrear objetos numa cena é normalmente dividido em duas partes: a primeira parte consiste em localizar o objeto a ser rastreado e descrevê-lo de maneira que ele possa ser encontrado nos frames seguintes; a segunda parte consiste em rastrear o objeto prevendo a posição que ele estará nos próximos frames e poder medir se a previsão está de acordo com a posição real do objeto.

Para a localização e caracterização de objetos, existem diversos algoritmos consagrados pela literatura, como segmentação, agrupamento e correspondência com modelo.

Já para rastrear objetos, diversos tipos de algoritmos surgiram para cumprir essa função. Por exemplo, os filtros adaptativos são amplamente usados para solucionar o problema da estimativa das posições futuras.

Assim sendo, o objetivo desse trabalho é desenvolver um método automático de localização e rastreamento da bola de futebol, que exija o mínimo de intervenções humanas no processo. O método consiste em um sistema alternante, que primeiramente localiza a bola a partir de nenhuma informação prévia sobre o

vídeo. Depois de encontrá-la, o sistema alterna para o método de rastreio que, utilizando as informações geradas pelo localizador, prevê o estado futuro da bola, facilitando encontrá-la nos frames seguintes.

Entretanto, o algoritmo deve ser capaz de contornar diversas dificuldades, como o fato da bola ser muito pequena em relação ao tamanho total da imagem, frequentemente sofrer oclusão durante as jogadas e, quando em alta velocidade, ter sua imagem severamente borrada.

A Figura 1 mostra um caso típico de oclusão total, onde a bola não pode ser encontrada nem por um observador humano. Ela se encontra atrás do pé do jogador de azul.

Já a Figura 2 mostra um caso de oclusão parcial, onde a bola está parcialmente escondida atrás da luva do goleiro, se confundindo com ela.

Na Figura 3 é apresentado um caso de borramento, onde a bola fica distorcida, sendo difícil de localizá-la.

Figura 1 – Exemplo de Oclusão total

Para contornar esses problemas será utilizada segmentação e o algoritmo de Viterbi para a localização da bola, e filtro de partículas para rastreá-la, que possui grande robustez frente a oclusões.

Por fim, algumas premissas foram estabelecidas. O algoritmo considera que a seqüência de vídeo é obtida a partir de apenas uma câmera, ou seja, não há descontinuidades no fluxo de vídeo. A mudança no iluminante entre vídeos, ou mesmo entre os frames do mesmo vídeo pode acarretar em erros no cálculo da segmentação, já que variações na cor e intensidade do iluminante resultam em variações nas cores e intensidades dos objetos da cena. Por isso, o iluminante é considerado constante durante a execução e uniforme sobre toda a cena, apesar de o algoritmo apresentar certa robustez com relação à variação das características do iluminante na cena. O algoritmo leva em consideração a cor da bola utilizada na partida, portanto precisa ser iniciado com a cor da bola que foi utilizada no jogo. No caso deste trabalho, a bola foi considerada branca e os cálculos apresentados são

Figura 2 – Exemplo de Oclusão Parcial

baseados nessa premissa.

Esse trabalho foi dividido da seguinte forma: O capítulo 2 apresenta o estado da arte na área de rastreamento de objetos em vídeos. No capítulo 3, é mostrado as etapas de pré-processamento do vídeo, que é convertido para um formato padrão para ser processado pelo algoritmo. O capítulo 4 descreve detalhadamente o algoritmo proposto, bem como as alternativas propostas para comparação. No capítulo 5, é explicado a maneira como foi implementado o algoritmo, sua subdivisão e organização hierárquica. O capítulo 6 apresenta os resultados dos testes comparativos entre os métodos da literatura e o método proposto, bem como um comparativo entre as alternativas propostas no capítulo 4. E, por fim, o trabalho finaliza no capítulo 7 apresentando as conclusões e resultados finais.

Figura 3 – Exemplo de Borramento

2 REVISÃO BIBLIOGRÁFICA

A localização de objetos é amplamente estudada desde os primórdios do processamento de imagens. Para tanto diversos algoritmos são utilizados para se distinguir entre o que é descartado, chamado de fundo, e os objetos de interesse.

Em vídeos esportivos não é diferente, e algumas alternativas podem ser citadas. Os autores (ISHII; et. al., 2007) utilizam diferença entre quadros para detectar a bola, baseado no fato de que a bola está em movimento na grande maioria dos casos, e que a câmera não possui um movimento muito grande, o que levaria a gerar erros na medida. Por sua vez, (SEO; et. al., 1997) utiliza máscaras como modelos de jogadores para localizá-los, entretanto a bola é inicializada manualmente. Outro exemplo é o de (XU; et. al., 2005) que adquire previamente imagens do campo e as utilizam como plano de fundo. Assim ele localiza os objetos por subtração de cada frame com o fundo obtido. Outra solução interessante é de (YU; et. al., 2003) que detecta a bola removendo o gramado por um algoritmo de crescimento da região por agrupamento de pixels, usando como característica principal o fato de a grama possuir cor aproximadamente constante em todo o campo. Já (LIU; et. al., 2006), utiliza segmentação por cor para separar a bola do fundo.

Da mesma forma que a localização de objetos em vídeos evoluiu nas últimas décadas, o rastreamento de objetos teve significativos avanços. Os filtros adaptativos são muito utilizados no rastreamento de objetos. Por exemplo, o Filtro de Kalman, que é muito bem explicado em (WELCH; et al., 2006), é muito eficiente e pode ser utilizado, inclusive, para rastreamento de objetos em tempo real (PIOVOSO; et al., 2003). Os autores (LIU; et. al., 2006), (SEO; et. al., 1997) utilizam tal filtro para rastrear a bola.

O filtro de partículas é utilizado para rastrear objetos, e tem seu uso em vídeos esportivos sendo muito difundido ultimamente. (OK; et al., 2002) e (NUMMIARO; et al.,2003) rastreiam os jogadores auxílio desse método, inclusive sob forte oclusão.

Ambos os sistemas de rastreamento precisam ser auxilados por uma função que precisará verificar o quão bom ficou a sua projeção. Os autores (SEO; et. al., 1997) utiliza modelos para verificar se o valor encontrado pelo Filtro de Kalman está correto, enquanto (LIU; et. al., 2006) utiliza correlação cruzada para verificar a semelhança entre candidatos. Os autores (NUMMIARO; et al.,2003) propõe uma nova técnica utilizando semelhança entre os histogramas, que é medido pela distância de Bhattacharyya.

O problema da oclusão também já foi abordada por diversos autores. A solução mais comum é utilizar diversas câmeras para rastrear a bola. Podemos citar como exemplo (XU; et. al., 2005) utiliza diversas câmeras sincronizadas para localizar e rastrear a bola e os jogadores. Essa solução, apesar de apresentar robustez elevada, possui um alto custo e grande complexidade de implementação. Assim alguns autores propuseram algoritmos capazes de rastrear objetos em vídeos obtidos de apenas uma câmera, porém com robustez a oclusões. Em (LIU; et. al., 2006) os autores estimam a posição e o movimento de jogadores e da bola no espaço tridimensional a partir do vídeo obtido de uma transmissão comum de televisão.

3 PRÉ-PROCESSAMENTO

As imagens obtidas da transmissão dos jogos vêm normalmente codificadas em algum padrão de vídeo. Para que o algoritmo possa ser executado, o vídeo precisa ser decodificado. Para isso foi criada uma etapa de pré-processamento que, embora seja executado ao mesmo tempo do que o algoritmo de rastreamento, é independente do mesmo.

Para a decodificação do vídeo foi utilizada a API do Directx fornecida pela Microsoft, chamada Microsoft SDK, de distribuição livre. Dessa forma foi possível decodificar dentro da própria aplicação utilizando a lista de *codecs* presente no computador.

Além de decodificar o vídeo, o pré-processamento é responsável pela conversão dos frames para o espaço de cores WXY. Esse espaço foi escolhido porque separa o canal de intensidade dos canais de cores, além de permitir o cálculo de distância cromática utilizando apenas a distância euclidiana.

Essa propriedade é importante, pois, mesmo que a bola seja branca, se o iluminante da cena não for igualmente branco, a cor apresentada pela bola na imagem terá um desvio em direção à cor do iluminante, o que exige uma compensação do algoritmo, essa compensação é mostrada no capítulo 4.1.1.1.

O espaço apresenta outras vantagens com relação aos outros espaços de cores encontrados na literatura. O espaço RGB não possui separação entre a intensidade e a informação de cor. YCbCr necessita utilizar cálculos muito mais complexos para calcular cada um dos canais, enquanto o HSI utiliza funções trigonométricas para calcular seus canais. Já o espaço WXY é de rápida conversão a partir dos canais RGB provenientes da imagem original, pois a conversão é feita utilizando-se apenas números inteiros.

O espaço de cores WXY foi primeiramente proposto pelo artigo em anexo e desenvolvido no LAPSI. Ele está disponível na biblioteca Lili2, que foi utilizada na implementação do algoritmo.

4 O ALGORITMO

O algoritmo é dividido em duas partes conforme pode ser visto na Figura 4. Cada uma dessas partes é subdividida em outras duas e é feita uma tomada de decisão. Na primeira parte, chamada de detecção da bola, localiza-se a bola a partir de toda a imagem, sem nenhuma informação prévia sobre sua posição. Nessa parte utiliza-se um processo de segmentação por cor, que busca identificar os pixels que possuem cor semelhante com a bola. Em seguida, esses pixels são agrupados em regiões com o auxílio de um algoritmo de agrupamento por crescimento de região e passam a ser tratados em conjunto, a partir desse momento são denominados candidatos.

Esses candidatos passam, então, por um processo de seleção por eliminação e por uma versão adaptada do algoritmo de viterbi.

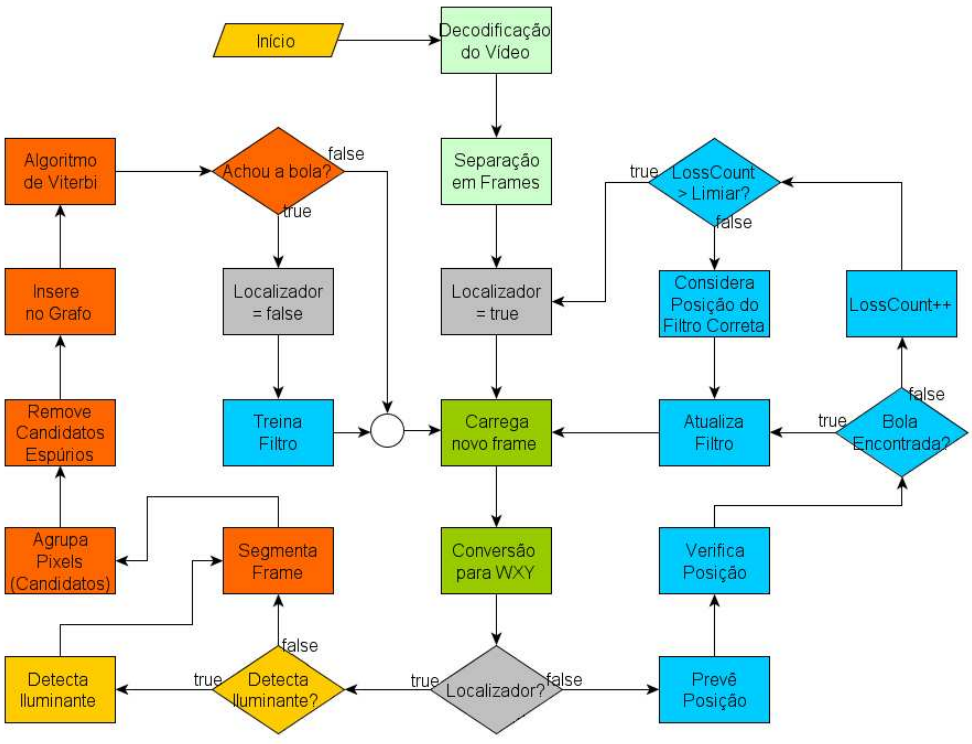


Figura 4 – Fluxograma do Algoritmo

Assim que a bola for encontrada, o processo alterna automaticamente para a segunda parte, chamada de Rastreamento da Bola, que utiliza o algoritmo de filtro de partículas. O filtro precisa ser auxiliado por um comparador, que permite avaliar se a posição indicada pelo filtro corresponde ao padrão da bola obtido do detector.

Em caso de perda do rastro da bola, o algoritmo retorna, então, para a etapa de detecção da bola, reiniciando o processo.

Essas etapas serão descritas abaixo.

4.1 DETECÇÃO DA BOLA

A detecção da bola é realizada em duas etapas. A primeira, descrita a seguir, acha as regiões dentro do quadro atual onde a bola pode estar localizada. Essa etapa é chamada Processamento Intraframe. A segunda etapa é a decisão sobre qual dessas regiões contém realmente a bola. Essa decisão é tomada considerando a relação das regiões dos frames anteriores com as do frame atual. Por esse motivo essa etapa é chamada de Processamento Interframe.

4.1.1 Processamento Intraframe

O vídeo, depois de pré-processado, é separado em frames e entra no algoritmo seqüencialmente. Como a iluminação varia de jogo para jogo, e às vezes dentro do mesmo jogo, o iluminante é estimado a cada cinco frames. Depois as regiões são identificadas através de um algoritmo de segmentação seguido por um algoritmo de crescimento de regiões. E, finalmente, aqueles candidatos que não se enquadram em alguns pré-requisitos básicos são descartados.

Os exemplos utilizados nesse capítulo terão como referência o frame original mostrado na Figura 5, retirado de um dos vídeos de testes.

4.1.1.1 Estimativa da cor e da intensidade do iluminante

As imagens podem ser consideradas como formadas por duas componentes: a quantidade de luz incidindo na cena a ser observada e a quantidade de luz refletida pelos objetos contidos na cena (GONZALEZ; et al., 2000), conforme é mostrado em (1).

$$f(x,y) = i(x,y)r(x,y) \quad (1)$$

Na equação, $i(x,y)$ representa o iluminante da cena, onde $0 < i(x,y) < \infty$; e

Figura 5 – Frame Original

$r(x, y)$ representa a refletância dos objetos da cena, onde $0 < r(x, y) < 1$.

Uma vez que foi considerado que a bola é branca, podemos considerar que ela apresentará intensidade e cor igual ou muito semelhante à intensidade e cor do iluminante. Dessa forma, podemos considerar que os pixels que possuírem maior refletância aparecerão na imagem com as características do iluminante. Assim, considerando que o iluminante não varia muito em toda a imagem, pode-se considerar a cor e intensidade dos pixels de maior intensidade na imagem como uma boa estimativa do iluminante. (EBNER, 2007)

Para determinar os pixels de maior intensidade utiliza-se o histograma cumulativo do canal W. O histograma cumulativo é calculado a partir do histograma da imagem, conforme explicam (GONZALEZ; et al., 2000) e (JAIN, 1989).

O histograma de uma imagem digital com L tons de cinza no intervalo

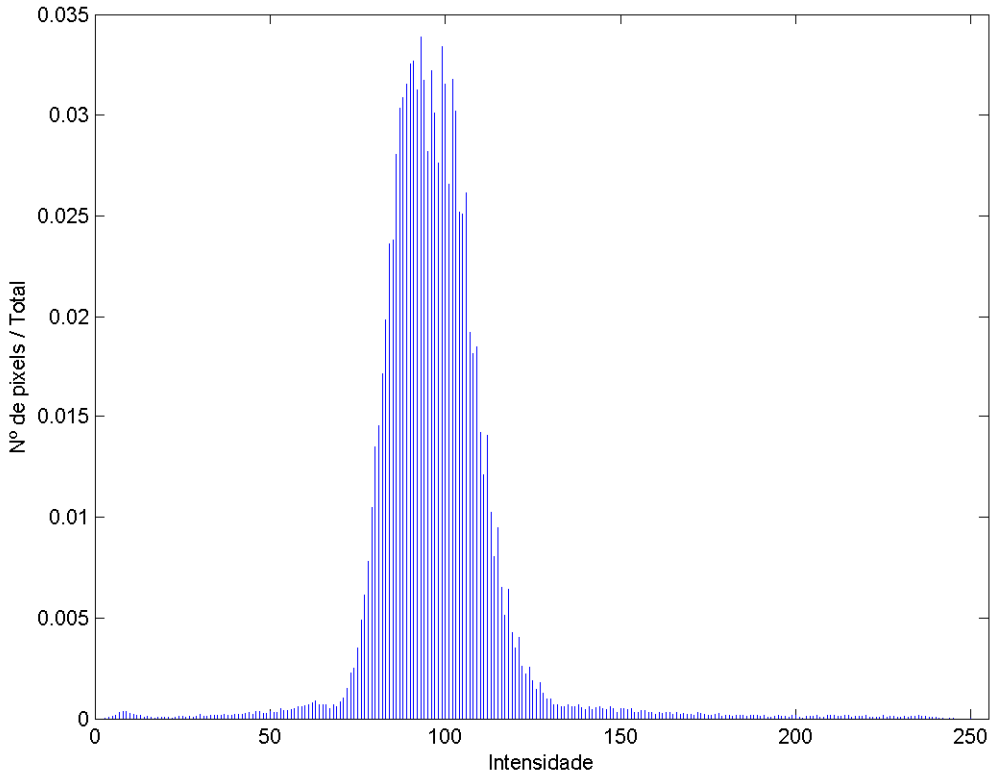


Figura 6 – Exemplo de Histograma

$[0, L-1]$ é uma função discreta. Conforme mostra (2).

$$hist(r_k) = n_k/N \tag{2}$$

onde r_k é o k-ézimo nível de cinza, n_k é a quantidade de pixels na imagem com esse nível de tom de cinza, N é o número total de pixels da imagem e $k = 0, 1, \ldots, L-1$ (GONZALEZ; et al., 2000). A Figura 6 mostra o histograma calculado à partir do canal W da Figura 5.

Como já mencionado acima, o histograma cumulativo é construído a partir do histograma da imagem e, para cada tom de cinza, o resultado corresponde à soma dos tons de cinza menores que ele. O histograma cumulativo correspondente ao histograma da Figura 6 é mostrado na Figura 7 e é calculado por (3).

$$hist_{cum}(\mathrm{r_k}) = \sum_{i=0}^{k} hist(r_i) \tag{3}$$

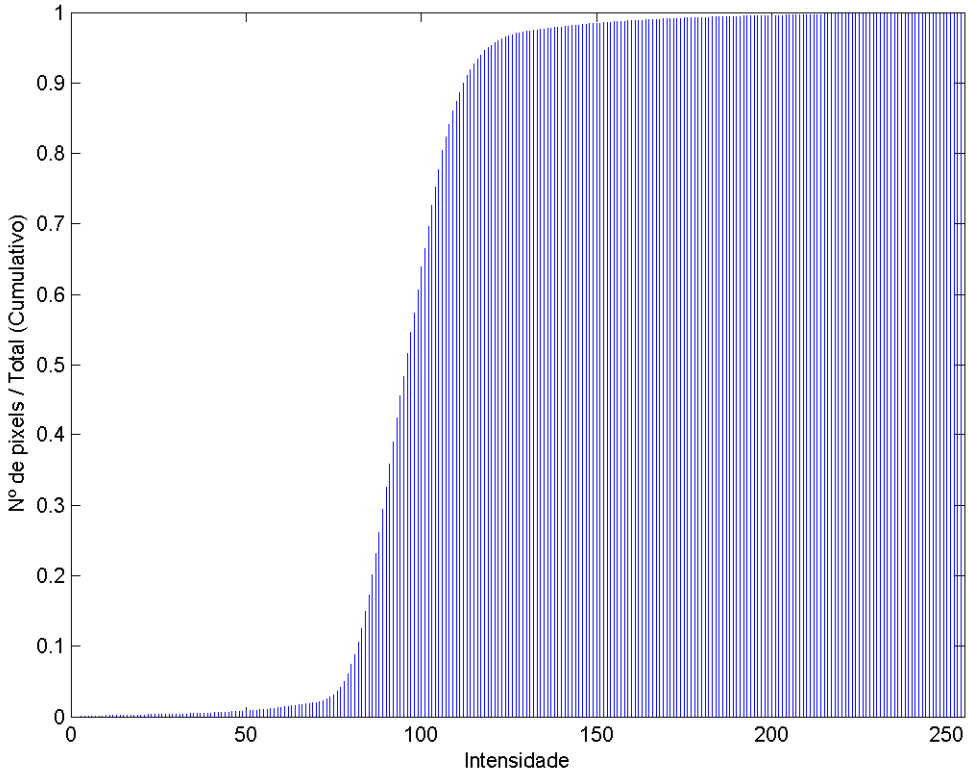


Figura 7 – Exemplo de Histograma Cumulativo

Os valores do histograma cumulativo ficam dentro do intervalo [0, 1].

Para se obter os pontos de maior intensidade, foram estabelecidos dois limiares, um baixo e outro alto. O limiar alto (L_h) é estabelecido na intensidade correspondente ao valor 0,99 no histograma cumulativo, enquanto o baixo (L_l) ao valor de intensidade que corresponde a 0,95 do histograma cumulativo. A Figura 8 mostra o histograma cumulativo, com as linhas pontilhadas em magenta e ciano em 0,99 e 0,95 respectivamente. A linha vermelha representa o valor calculado de L_h, enquanto a linha verde representa L_l.

Os valores dos limiares são empíricos e foram obtidos através de experimentos realizados com diversos vídeos.

Assim que é encontrado o valor dos limiares é calculada a média das cores dos pixels cujo canal W tem valor maior que L_h. Como os pixels com valor acima de

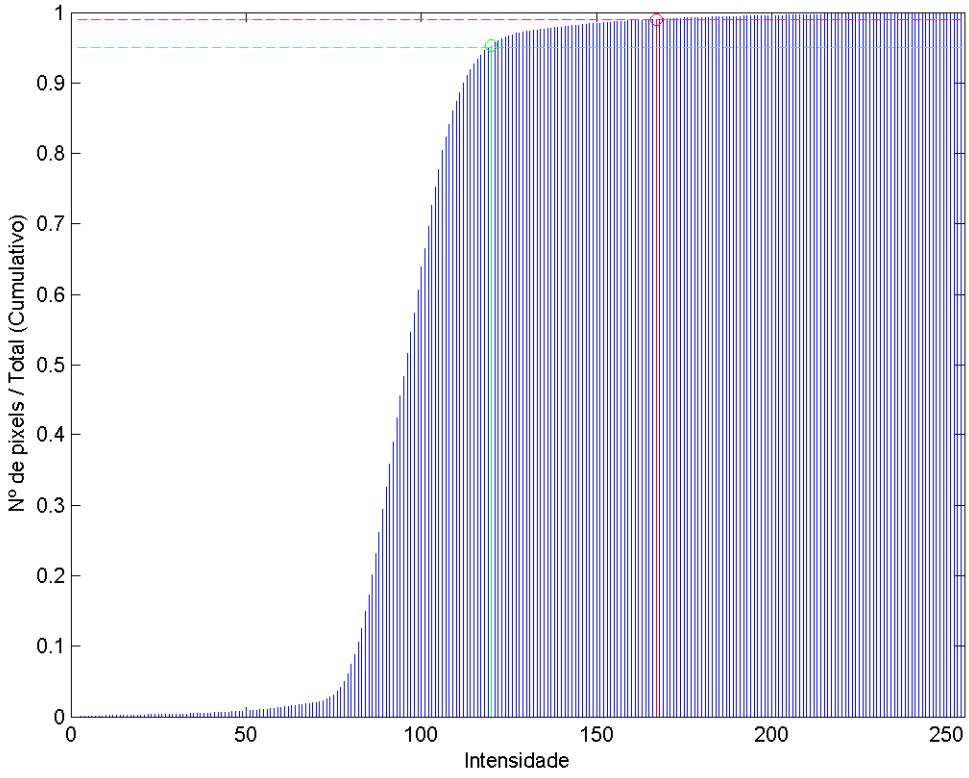


Figura 8 – Cálculo dos Limiares a partir do Histograma Cumulativo

L_h são considerados amostras do iluminante, a cor média desses pixels pode representar, também, uma estimativa do valor médio da cor do iluminante. Essa estimativa da cor do iluminante é chamada *White-Patch Retinex* e é explicado por (EBNER, 2007).

A Figura 9 mostra os limiares (L_h e L_l) sobrepostos ao histograma da imagem de forma a mostrar a intensidade e a quantidade de pixels que serão utilizados para estimar o iluminante.

4.1.1.2 Segmentação

Depois de calculados os limiares, uma imagem em tons de cinza auxiliar é

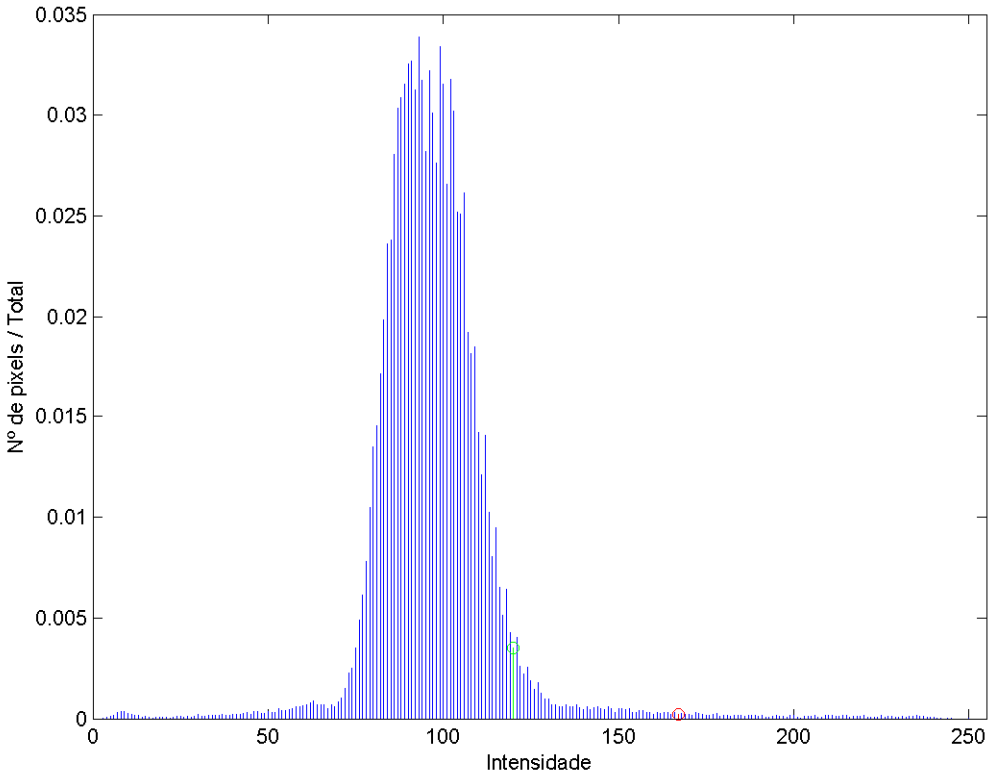


Figura 9 – Limiares calculados apresentados no Histograma

criada com o mesmo tamanho da imagem original. Cada pixel dessa imagem recebe seu valor conforme o valor do pixel de mesma posição na imagem original.

Para cada pixel é calculada a distância cromática entre o valor de cor do pixel (canais X e Y) e o valor de cor do iluminante.

Caso o valor do pixel possua distância cromática superior a um limiar $dist_{cor}$ ou possua o valor de W inferior a L_l, o pixel recebe zero. Os pixels restantes são divididos em dois grupos: aqueles que possuírem W superior ou igual à L_h recebem o valor correspondente ao branco (255), os restantes recebem o valor de cinza (128).

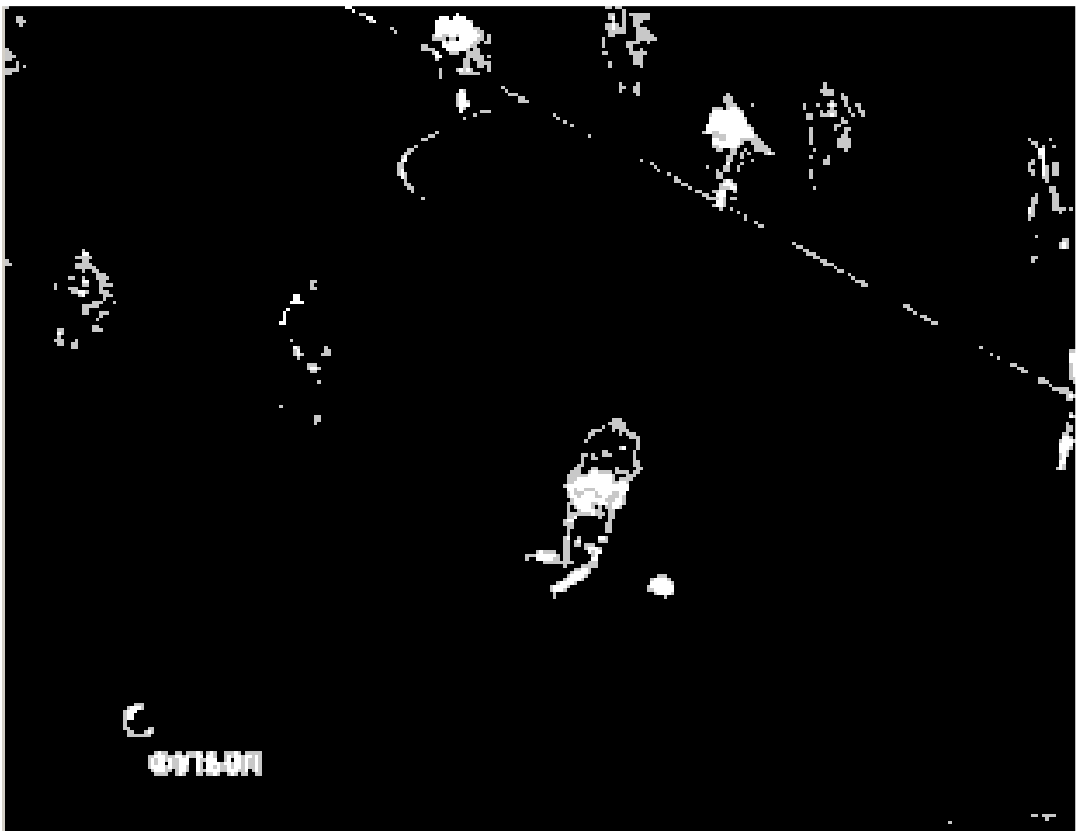


Figura 10 – Frame após segmentação

O limiar $dist_{cor}$ é aplicado para que se garanta que a cor dos pixels segmentados possua uma semelhança mínima com a cor da bola, no caso branca.

Ele pode ser modificado na inicialização do algoritmo e tem seu valor inicial empiricamente determinado em 200.

A imagem resultante dessa etapa é chamada de imagem segmentada, e pode ser vista na Figura 10.

4.1.1.3 Algoritmo de crescimento de regiões por agregação de pixels

Conforme explica (GONZALEZ; et al., 2000), o algoritmo de crescimento de regiões por agregação de pixels, aqui chamado apenas por algoritmo de crescimento, agrupa os vizinhos provocando o crescimento de regiões homogêneas. Cada ponto branco, chamado de semente, é um ponto de partida para o algoritmo e é considerado como uma região com tamanho igual a um pixel. A seguir, os pontos adjacentes que forem brancos ou cinza são adicionados à região e marcados como visitados. Para cada ponto adicionado, o algoritmo inclui, recursivamente, os pontos brancos ou cinzas adjacentes a ele, inclusive fundindo regiões que estejam lado a lado. Esse processo é executado até que não haja mais pontos para serem adicionados às regiões.

O algoritmo marca, ainda, os pontos pertencentes a uma região e que forem adjacentes a pontos pretos como sendo pontos de borda.

Assim, ao final da execução da função, todos os pontos brancos ou cinzas que estiverem ligados direta ou indiretamente a um ponto branco inicial estarão dentro de uma mesma região, que é chamado então de candidato à bola. Os pixels cinza que não se encontrarem em nenhuma região e os pretos serão classificados como fundo e descartados.

4.1.1.4 Caracterização dos candidatos

Depois de segmentada e agrupada em regiões, a imagem contém apenas os candidatos. Cada um desses candidatos à bola é uma região possível de conter a localização verdadeira da bola. A partir daí, são extraídas diversas informações dos candidatos para verificar qual deles corresponde à bola.

Junto com o algoritmo de crescimento, é somado o número total de pixels que compõem o candidato, além dos quatro atributos de posição: os limites superior, inferior, esquerdo e direito. Com as essas informações, podemos calcular o mínimo retângulo que circunda o candidato (MBR), conforme mostrado na Figura 11. O MBR é representado por dois pontos: superior-direito e inferior-esquerdo.

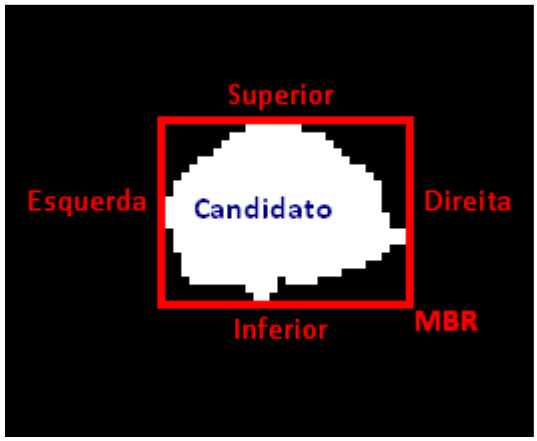


Figura 11 – Atributos do MBR

Nessa etapa é realizado um processo de seleção por eliminação. O procedimento visa diminuir os cálculos posteriores, eliminando o ruído proveniente da segmentação e os candidatos que correspondem a objetos que podemos afirmar, com certeza, que não são a bola.

Os critérios para eliminação de candidatos são descritos abaixo.

Tamanho Máximo (Figura 12): O tamanho do candidato é maior que o limiar $Tam_{máx}$. Visa eliminar candidatos que por ventura possam vir a existir a partir das camisas de jogadores, placas publicitárias ou linhas do campo.

Figura 12 – Tamanho Máximo

Tamanho mínimo (Figura 13): O tamanho do candidato é menor que o limiar $Tam_{mín}$. Tem como objetivo eliminar ruído proveniente da segmentação ou objetos muito pequenos.

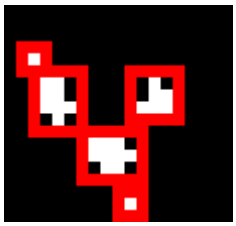


Figura 13 – Tamanho Mínimo

Relação altura-largura mínima (Figura 14): A relação entre a altura e a largura do MBR do candidato é maior que o limiar R_{al}. Elimina candidatos que possam ter sido considerados a partir de segmentos de linha de marcação do campo, meias e outros objetos que possuam altura desproporcional à largura.

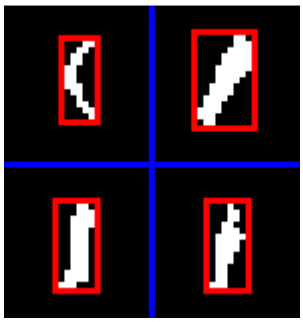


Figura 14 – Relação Altura Largura

Preenchimento mínimo (Figura 15): A relação entre a área do candidato e a área do retângulo circundante for menor do que o limiar R_{aa}. Visa eliminar linhas do campo e outros candidatos que possam ser vazados.

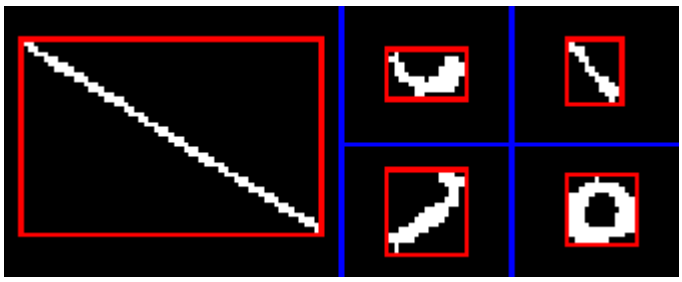


Figura 15 – Preenchimento do MBR

Marca d'água (Figura 16): O candidato pertence a uma região que foi previamente marcada como de marca d'água. Essa seleção visa eliminar placares, marcas d'água da emissora e outras propagandas que possam confundir o algoritmo de reconhecimento.

Figura 16 – Marca d'água

Os limiares podem ser definidos na inicialização do vídeo conforme as características e do posicionamento da câmera. Para os testes realizados, os valores foram definidos empiricamente, com base nos experimentos, de maneira que apenas os candidatos mais grosseiros são removidos, garantindo, assim, que a bola nunca fosse descartada nessa etapa. $Tam_{Máx}$ possui valor relativo ao tamanho da imagem e valor típico de 1/1000. $Tam_{mín}$ possui valor típico de 20 pixels. Já R_{al} foi definido para 1,5 e R_{aa} para 0,5, com base no que sugere (LIU; et. al., 2006).

A Figura 17 mostra o frame apresentado anteriormente, depois de que os candidatos foram identificados. Os candidatos em vermelho são aqueles considerados válidos, em lilás os eliminados por tamanho, em verde, os eliminados por relação entre altura e largura, em azul, os eliminados por pouco preenchimento do MBR, em amarelo, os muito pequenos e em marrom, os eliminados por

Figura 17 – Frame depois do processamento Intraframe

pertencerem a uma região de marca d'água.

Também são calculadas algumas informações que serão de relevância para a seqüência de execução do algoritmo, principalmente para a etapa de Processamento Interframe. Calcula-se o centróide do candidato, conforme (4).

$$\mu_i = \left(\frac{1}{N} \sum_{n=0}^{N} x_n , \frac{1}{N} \sum_{n=0}^{N} y_n \right) \tag{4}$$

A partir do ponto central e dos pixels marcados como borda pelo algoritmo de crescimento de regiões, a distância média dos pixels de borda ao centróide é calculada conforme a (5).

$$\eta_i = \frac{1}{M} \sum_{k=0}^{M} (\|p_k - \mu_i\|) \tag{5}$$

Com esses dados é possível, então, calcular a variância do raio do candidato, conforme mostra (7). Para tanto, calcula-se a raiz quadrada da soma dos quadrados das diferenças entre a distância do pixel de borda, obtidos anteriormente, e a distância média dos pixels de borda ao pixel central. Com o valor da variância, pode-se, finalmente, estimar a semelhança de um candidato com a figura de um círculo através da (6).

$$v_i = \begin{cases} 1 - \sqrt{c_i} & c_i \leq 1 \\ 0 & c_i > 1 \end{cases} \tag{6}$$

onde,

$$c_i = \frac{1}{M \mu_r^2} \sum_{k=0}^{M} (\|p_k - \mu_i\| - \eta_i)^2 \tag{7}$$

Em (7), c_i é chamado Variância Circular do i-ézimo candidato. Quanto menor for o valor de c_i, mais o contorno do candidato se parece com o contorno de um círculo. p_k é um ponto de borda, M é o número de pontos de contorno, μ_i é o

centróide do candidato, calculado em (5), η_i é a média das distâncias dos pontos de contorno ao centróide.

O procedimento é repetido para todos os candidatos obtidos na segmentação, gerando uma lista de listas encadeadas com todos os possíveis candidatos a bola. Cada candidato recebe um código de referência e é armazenado. Para cada um deles é calculado uma série de informações que serão utilizadas durante a execução do algoritmo. Todas as informações calculadas são armazenadas com o intuito de poderem ser livremente utilizadas nos próximos passos do algoritmo.

4.1.2 Processamento Interframe

Após processar a imagem no âmbito do frame atual do vídeo, se inicia uma etapa de processamento entre os frames do vídeo. Nessa etapa é considerado o fato de a bola permanecer com suas propriedades entre os frames. Mesmo que os outros candidatos se pareçam com uma bola em um frame, eles não possuem a tendência a continuar parecendo-se com a mesma em diversos frames seguidos, sob pena de serem confundidos mesmo por observadores humanos. Assim, a bola será considerada aquela que melhor preencher os pré-requisitos do algoritmo por uma seqüência mínima de frames.

Conforme sugere (LIU; et. al., 2006), os candidatos detectados são utilizados para montar um grafo ponderado. Conforme pode ser visto na Figura 18, cada nodo do grafo – em azul – representa um candidato; recebe um peso igual à semelhança do candidato com um círculo – valor interno ao nodo –, que foi calculada em (6). Cada nodo de um frame é ligado com os nodos do frame anterior através de uma borda que possui um peso que representa a semelhança entre ambos – valor sobre

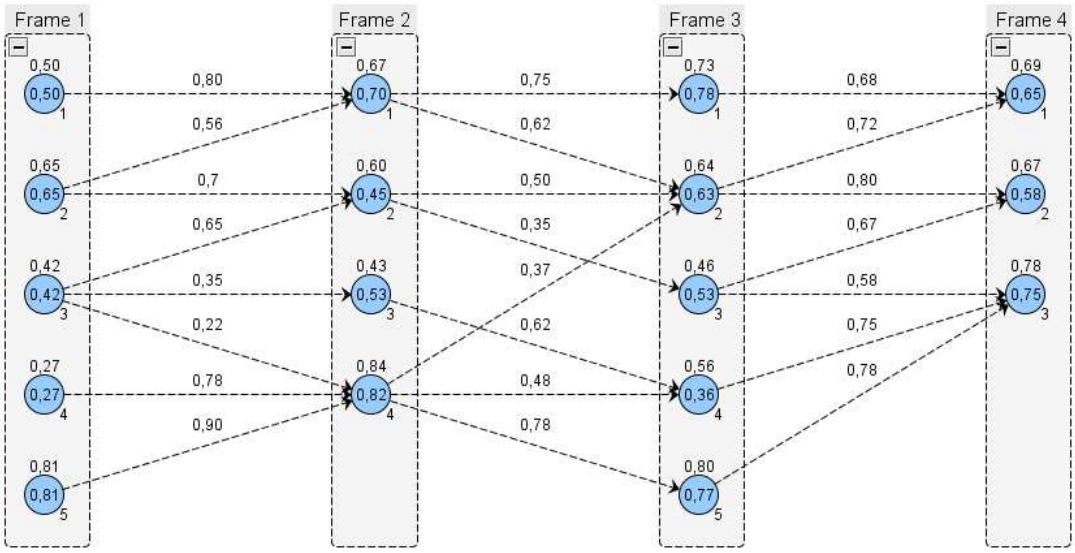


Figura 18 – Grafo criado a partir dos candidatos

a borda –, que pode ser calculada em (8), conforme sugere (LIU; et. al., 2006). Como a localização da bola não se altera muito entre um frame e outro, apenas os pares de candidatos que possuírem distâncias menores que um limiar $d_{máx}$ são considerados para designar as bordas do grafo.

$$e_{i\,j}^{t} = \frac{\left(\omega_s \zeta_{i\,j}^{t} + \omega_g g_{i\,j}^{t}\right)}{\sqrt{1 + \left(d_{i\,j}^{t} / d_{máx}\right)^2}} \tag{8}$$

Na equação acima, o índice t índica o frame atual; *i* e *j* indicam o *i*-ézimo candidato do frame t e o *j*-ézimo candidato do frame $t-1$, respectivamente. $\zeta_{i\,j}^{t}$ e $g_{i\,j}^{t}$ são as semelhanças de tamanho e tons de cinza, respectivamente. ω_s e ω_g são os pesos correspondentes e foram fixados em 0,5. O valor $d_{i\,j}^{t}$ é a distância euclidiana entre a posição do candidato i no frame t e a posição do candidato j no frame $t-1$.

Ainda nessa equação, podemos definir $\zeta_{i\,j}^{t}$ pela (9). Nesse caso, Δh e Δw são a diferença de altura e largura entre os MBRs dos dois candidatos, respectivamente. Conforme (LIU; et. al., 2006), pode-se considerar, que $(\Delta h, \Delta w)$ obedece a uma

distribuição gaussiana com média zero $\mathrm{N}_{\Delta}(0,\Sigma)$, onde Σ é a variância dessa distribuição e foi previamente estimado por amostras conhecidas da bola. Já $g_{i\,j}^{t}$é definido em (10) e calcula a correlação cruzada entre dois candidatos em tons de cinza. $W_i(k)$ e $W_j(k)$ correspondem ao valor do canal W no pixel k dos candidatos i e j respectivamente. O método pressupõe que ambos os candidatos possuem altura e largura idênticas. Caso isso não se confirme, é necessário realizar uma etapa de normalização de tamanho do candidato menor. Essa etapa consiste em ampliar a região amostrada da imagem original do candidato menor de maneira a ampliá-lo até atinja as dimensões iguais àquelas do candidato maior.

$$\zeta_{i\,j}^{t} = \frac{1}{\sqrt{2\pi\Sigma^2}} e^{-\left(\frac{\Delta h^2+\Delta w^2}{2\Sigma^2}\right)} \tag{9}$$

$$g_{i\,j}^{t} = \frac{\sum_k W_i\,(k) W_{\mathrm{j}}(k)}{\sqrt{\sum_k W_{\mathrm{i}}(k) W_{\mathrm{i}}(k)}\sqrt{\sum_k W_{\mathrm{j}}(k) W_{\mathrm{j}}(k)}} \tag{10}$$

Os valores acima dos nodos são os pesos calculados pelo algoritmo de Viterbi e são explicados a seguir.

4.1.2.1 Algoritmo de Viterbi

O algoritmo de viterbi é comumente utilizado numa ampla gama de aplicações de comunicação e de armazenamento de dados. É utilizado para decodificar códigos convolutivos, em detecção da banda base em sistemas sem fio e também para a detecção de dados gravados em discos magnéticos (RABAEY; et al., 2003).

O algoritmo é mostrado, normalmente, como um diagrama de Trellis, que é um diagrama de estados indexado no tempo. A detecção de máxima semelhança

pode ser descrita como o caminho mais provável dentro de um diagrama de Trellis (RABAEY; et al., 2003).

O grafo montado no capítulo anterior, mostrado na Figura 18, corresponde a tal diagrama, em que cada nodo corresponde a um estado em um determinado tempo, e cada ramo representa a transição para um novo estado no próximo instante de tempo (FORNEY, 1973).

Para decidir qual candidato representa efetivamente à bola no jogo, o algoritmo de viterbi é utilizado conforme (LIU; et. al., 2006). Cada frame é representado por um passo no algoritmo e cada candidato é um nodo. O algoritmo utiliza o grafo ponderado, que foi construído iterativamente. A cada passo inserem-se os candidatos do frame atual e suas ligações, conforme explicado anteriormente.

O algoritmo é iniciado conforme (11)0. Para cada passo, o algoritmo é atualizado e cada nodo i recebe seu valor δ_i^t, que é calculado conforme (12)0 e é mostrado sobre o nodo na Figura 18 e na Figura 19, onde N_{t-1} é o número de nodos do passo anterior, cada um dos termos pode ser ponderado pelo seu respectivo

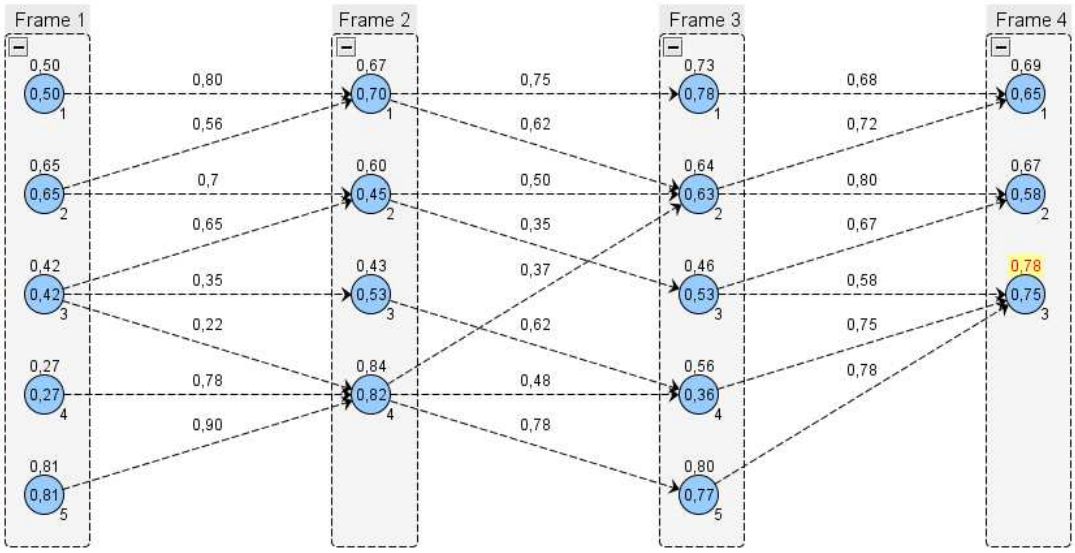


Figura 19 – Bola encontrada pelo algoritmo de Viterbi

peso $(\omega_\delta, \omega_e, \omega_v)$. Esses pesos podem ser determinados no início do algoritmo e nos testes foram todos fixados em $1/3$. O processo de localização da bola é executado, frame a frame, enquanto o caminho ótimo, calculado com auxílio de (13), possuir profundidade menor que um limiar T.

$$\delta_i^1 = v_i^1 \tag{11}$$

$$\delta_i^t = \max_{1 \le i \le N_{t-1}, d_{i,j}^{t-1} \le d_{max}} \left(\omega_\delta \delta_i^{t-1} + \omega_e e_{i,j}^{t-1} + \omega_v v_j^t\right) \tag{12}$$

$$\psi_t(j) = \arg\max_{1 \le i \le N_{t-1}, d_{i,j}^{t-1} \le d_{max}} \left(\delta_i^{t-1} + e_{i,j}^{t-1} + v_j^t\right) \tag{13}$$

Caso a profundidade do grafo do algoritmo de viterbi atinja um valor superior a um limiar T, a bola é considerada encontrada naquele ponto que possuir o maior valor no seu nodo na iteração atual. Caso dois nodos possuam valores iguais, aquele que possuir o maior valor de semelhança com a bola nesse frame é selecionado. Persistindo o empate, um candidato é sorteado.

A Figura 20 mostra o exemplo de recursão para se obter as posições anteriores mais prováveis de representarem efetivamente a bola. Esse dado é utilizado na inicialização do rastreador.

Nas figuras de exemplo, o limiar T utilizado foi 4, já no programa, o limiar T pode ser determinado no início da execução e nos testes realizados foi fixado em 5.

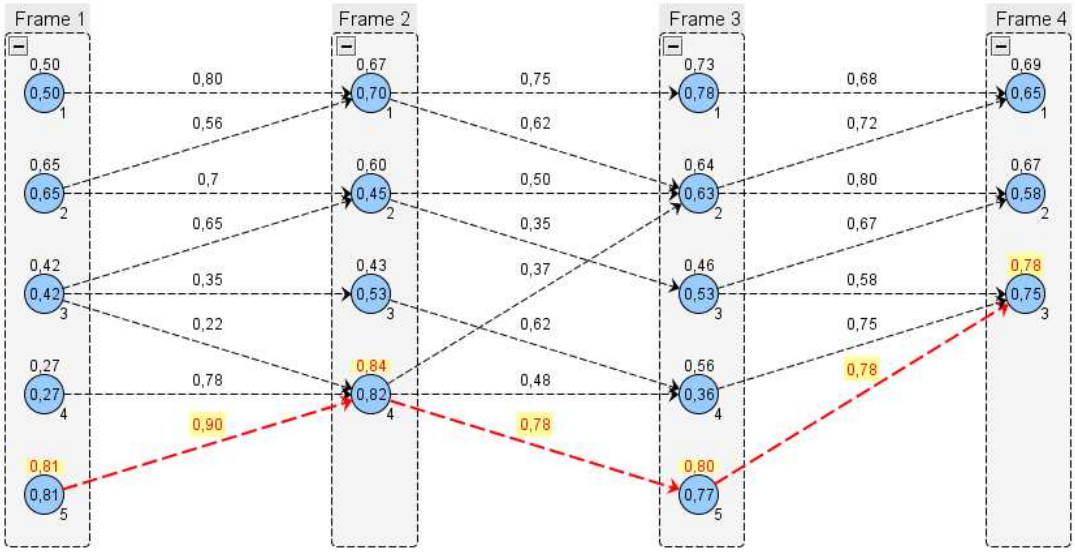


Figura 20 – Cálculo das posições anteriores da bola pela equação de recursão

4.2 RASTREADOR DA BOLA

Após a bola ser encontrada, na detecção pelo algoritmo de viterbi, é utilizado um filtro adaptativo para rastrear a bola. Nesse trabalho foi realizado um estudo comparativo entre o Filtro de Kalman baseado em detecção por semelhança de padrão e o Filtro de Partículas para se determinar qual tem melhor desempenho no caso de rastrear a bola.

Inicialmente é explicado o Filtro de Kalman, passando em seguida ao Filtro de Partículas. Os resultados dos experimentos são exibidos no capítulo 6.

Todas as informações necessárias para inicializar os filtros adaptativos são retiradas dos dados gerados pelo detector.

4.2.1 Filtro de Kalman

O filtro de Kalman resolve o problema de estimar o estado $x \in \mathbb{R}^n$ de um processo discreto que é governado por uma função estocástica linear, visto na (14) e a medição $z \in \mathbb{R}^m$, visto na (15) (WELCH; et al., 2006).

$$x_t = Ax_{t-1} + Bu_{t-1} + w_{t-1} \tag{14}$$

$$z_t = Hx_t + v_t \tag{15}$$

Pelo fato de apenas observarmos o sistema a ser previsto, u é sempre igual a zero e (14) possui apenas o primeiro e o último termo diferentes de zero. Nessa equação, A representa a relação entre o estado anterior, medido e o estado previsto. Já na equação (15), H representa uma matrix m x n representando a relação entre o estado previsto (x_t) e o valor medido (z_t). As variáveis randômicas w_t e v_t são chamadas de ruído de processo e de medição, respectivamente. Nesse trabalho,

serão consideradas como ruído branco, independentes entre si e com distribuição normal. Assim a distribuição é dada conforme (16).

$$\begin{aligned} p(w_t) &= N(0, Q) \\ p(v_t) &= N(0, R) \end{aligned} \tag{16}$$

A covariância do ruído do processo (Q) e a covariância do ruído de medição (R) podem variar durante o tempo, porém elas serão consideradas constantes para nosso processo. Elas foram obtidas empiricamente através de testes de predição e medição.

O filtro utiliza um método de controle retroalimentado: ele estima o processo em algum momento e obtém a retroalimentação medindo o sistema ruidoso (WELCH; et al., 2006). Assim pode-se dizer que o filtro funciona em um processo de duas etapas: uma de previsão e outra de correção. Na etapa de previsão, mostrada em (17) e (18), o próximo estado do filtro é previsto a partir do estado atual. Essa previsão é chamada de estado *a priori*.

$$\hat{x}_t^- = A\hat{x}_{t-1} + Bu_{t-1} \tag{17}$$

$$P_t^- = AP_{t-1}A^T + Q \tag{18}$$

Em (18), P_t^- significa a estimativa da covariância do erro à *priori*. Ela é calculada a partir da covariância do erro a *posteriori*, que é realizada em (21) e da covariância do ruído do processo (Q). Como se pode notar, a equação é recursiva e deve ser inicializada no primeiro momento. Assim, P_0 possui valor constante, e segundo (WELCH; et al., 2006) pode ser inicializado com qualquer valor diferente de zero. No caso dessa dissertação, ele foi inicializado com a matriz identidade.

Depois de prever o próximo estado, o filtro realiza a correção. Nessa etapa, o valor medido é incorporado ao sistema para melhorar o estado *a priori*, criando um estado aprimorado *a posteriori*. A diferença entre o previsto e o medido é utilizada para aperfeiçoar as variáveis do filtro, diminuindo, assim, o erro de predição. Isso é

feito através do ganho K_t, obtido através de (19), que é calculado a partir da covariância do erro (P_t^-) e da covariância do ruído de medição (R) e também é conhecida como equação de Riccati. Esse ganho é multiplicado, então, à $z_t - H\hat{x}_t^-$, que é a diferença entre o estado estimado e o medido. As equações abaixo mostram a etapa de correção.

$$K_t = P_t^- H^T (H P_t^- H^T + R)^{-1} \quad (19)$$

$$\hat{x}_t = \hat{x}_t^- + K_t(z_t - H\hat{x}_t^-) \quad (20)$$

$$P_t = (I - K_t H) P_t^- \quad (21)$$

Na etapa de correção, calcula-se primeiramente o ganho de Kalman, K_t, mostrado em (19). Na sequência, mede-se o sistema para obter o seu estado atual (z_t), e então gerar uma estimativa da posição da bola *a posteriori* ($\hat{x}_t$), incorporando a medição, como mostra (20). Finalmente, se obtém a estimativa *a posteriori* da covariância do erro através de (21).

A cada rodada de cálculo, são utilizados os valores calculados *a posteriori* para prever os novos dados *a priori*. Essa propriedade recursiva torna o filtro de fácil implementação (WELCH; et al., 2006).

Para prever a posição da bola, foi utilizado o modelo de primeira ordem, sugerido por (LIU; et. al., 2006), abaixo:

$$X = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}; \quad Z = \begin{bmatrix} x \\ y \end{bmatrix}; \quad A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \quad (22)$$

Em (22), (x, y) representa o centro da bola, $(\dot{x}, \dot{y})$ representam a velocidade da bola. Conforme sugere (LIU; et. al., 2006), consideramos que a bola possui velocidade constante no plano da imagem. A matriz A acima está de acordo com essa premissa, uma vez que ao prever o estado seguinte do sistema, a velocidade,

permanece constante. A posição é atualizada acrescendo seu valor apenas o valor de velocidade.

Os resultados dos testes realizados com o filtro de Kalman podem ser vistos no capitulo 6.

4.2.2 Filtro de Partículas

O filtro de partículas foi utilizado como segunda alternativa no rastreamento da bola. Sua explicação e embasamento teórico são como segue.

O filtro de partículas foi desenvolvido originalmente para rastrear objetos em ambientes ruidosos (NUMMIARO; et al.,2003). Uma das maiores vantagens do algoritmo está em sua simplicidade, comparado com o filtro de Kalman, além de sua generalidade. Podemos atribuir a sua simplicidade à ausência da equação de Riccati que é utilizada no filtro de Kalman para calcular o ganho K_t (19). Ela é relativamente complexa computacionalmente, pois apresenta necessidade de cálculo de matrizes inversas. O filtro de partículas lida com a variabilidade com o uso de amostragem, repetindo a computação de uma fórmula de propagação relativamente simples (ISARD; et al., 1998).

A idéia chave do filtro de partículas é aproximar a distribuição de probabilidades por um grupo de amostras ponderadas $S = \{(s^{(n)}, \pi^{(n)}) | n = 1 \dots N\}$. Onde cada amostra *s* representa um estado hipotético do objeto, com uma probabilidade amostral correspondente π, sendo $\sum_{n=1}^{N} \pi^{(n)} = 1$. No caso em estudo, cada amostra é uma tentativa de acertar a posição da bola.

O estado do objeto rastreado é descrito pelo vetor X_t enquanto Z_t denota as respectivas observações. A evolução do grupo de amostras é descrito pela

propagação de cada amostra de acordo com o modelo do sistema. Cada elemento do grupo é então ponderado conforme os dados gerados das observações e então *N* amostras são sorteadas para repor as amostras atuas. A probabilidade de cada amostra de ser sorteada é dada por $\pi^{(n)} = p\left(z_t \middle| X_t = s_t^{(n)}\right)$. O estado médio do objeto é estimado a cada passo por (23).

$$E[S] = \sum_{n=1}^{N} \pi^{(n)} s^{(n)} \tag{23}$$

O filtro de partículas apresenta um rastreamento robusto, ao mesmo tempo em que modela o sistema incertamente. Ele pode manter suas opções abertas e considerar múltiplos estados do objeto simultaneamente. Já que existe a chance de estados menos parecidos com o objeto permanecerem no processo de rastreamento, o filtro de partículas pode lidar com oclusões curtas (NUMMIARO; et al.,2003). Essa propriedade é interessante pelo fato de a bola fica oclusa em muitos momentos dos vídeos de futebol.

Cada uma das amostras representa um retângulo e é representado por $s = \{x, y, \dot{x}, \dot{y}, J_x, J_y, \dot{a}\}$, onde x, y representa a localização do seu centro, $\dot{x}, \dot{y}$ a sua velocidade de movimento, J_x, J_y a largura e a altura, respectivamente, e $\dot{a}$ representa a mudança de escala.

O grupo de amostras é propagado através de um modelo dinâmico

$$s_t = \mathcal{A} s_{t-1} + \mathcal{w}_{t-1} \tag{24}$$

onde $\mathcal{A}$ define a componente determinística do modelo e $\mathcal{w}_{t-1}$ é um vetor de variáveis randômicas gaussianas.

O sistema foi simplificado como sendo de primeira ordem, conforme mostra a equação abaixo:

$$\mathcal{A} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (25)$$

Em (25), nota-se que tanto a velocidade quanto a mudança de escala são consideradas constantes.

O resumo do algoritmo pode ser visto na Figura 21. Como o algoritmo é inicializado pelo localizador da bola, no instante $t = 0$ do rastreador a bola é conhecida. Dessa forma existe apenas uma partícula na posição da bola. Sua velocidade $\dot{x}, \dot{y}$ é estimada pelo caminho realizado pela bola nos frames anteriores.

Do grupo de amostras $\left\{s_{t-1}^{(n)}, \pi_{t-1}^{(n)}, cum_{t-1}^{(n)}\right\}, n = 1, \ldots, N$ no passo $t-1$ construa-se um novo grupo de amostras $\left\{s_t^{(n)}, \pi_t^{(n)}, cum_t^{(n)}\right\}, n = 1, \ldots, N$ para o passo t.

A n-ézima amostra das N novas amostras é construída como segue:

1. **Escolha** a amostra $s'^{(n)}_t$ como segue:
   a. Gere um número randômico $r \in [0,1]$, distribuído uniformemente.
   b. Ache o menor j tal que $cum_{t-1}^{(j)} \geq r$
   c. Atribua $s'^{(n)}_t = cum_{t-1}^{(j)}$.

2. **Propague** as amostras utilizando a modelagem do sistema
   $$s_t^{(n)} = \mathcal{A} s'^{(n)}_t + w_t$$
   sendo w_t um vetor de variáveis randômicas gaussianas

3. **Meça** e pese a nova posição em termos das características z_t medidas:
   $$\pi_t^{(n)} = p\left(z_t \middle| x_t = s_t^{(n)}\right)$$
   normalizando para que $\sum_n \pi_t^{(n)} = 1$ armazenando junto com a probabilidade cumulativa como $\left(s_t^{(n)}, \pi_t^{(n)}, cum_t^{(n)}\right)$, onde
   $$cum_t^{(0)} = 0,$$
   $$cum_t^{(n)} = cum_t^{(n-1)} + \pi_t^{(n)} (n = 1, \ldots, N)$$

Assim que as N amostras estiverem construídas: estime, se desejado, a posição rastreada no passo t como

$$E[S_t] = \sum_{n=1}^{N} \pi_t^{(n)} s_t^{(n)}$$

Figura 21 – Algoritmo de Filtro de Partículas

Cada amostra então é considerada como a área da bola ampliada de um fator percentual que pode ser configurado no início do algoritmo. Pelo mesmo motivo, inicializa-se $cum_0^{(n)} = cum_0^{(0)} = 1$, pois o filtro possui apenas uma partícula.

4.2.2.1 Função Semelhança

Outra parte muito importante no bom desempenho do filtro de partículas é a maneira como é definida a função $p\left(z_t \middle| x_t = s_t^{(n)}\right)$ que calcula a probabilidade de que a amostra $s_t^{(n)}$ represente realmente a bola. Para tanto se calcula primeiramente a semelhança entre a amostra e o modelo da bola. Esse modelo é retirado da própria imagem no momento que a bola foi localizada. A essa função de semelhança é dada o nome de $sem(im_{templ}, im)$.

Os autores (NUMMIARO; et al.,2003) propõem uma solução utilizando semelhança entre os histogramas da bola e da amostra, chamada de distância de Bhattacharyya. Para essa dissertação, além desta, foram estudadas outras formas de determinar essa probabilidade. Assim, algumas funções de semelhança entre imagens foram utilizadas nos testes. De (LIU; et. al., 2006), foi utilizada a semelhança em tons de cinza que media a semelhança entre candidatos na etapa de localização da bola. Também, a partir da imagem binarizada proveniente da segmentação, dois métodos foram estudados: (BALLARD; et al., 1982) sugere um método de cálculo de semelhança entre imagens calculando-se a semelhança entre as projeções nos eixos dos pontos segmentados; o sexto cenário é sugerido como a correlação cruzada em tons de cinza entre as imagens resultantes da segmentação como forma de detectar a presença ou não da bola.

Os métodos foram testados e avaliados. Seus resultados são mostrados na seção 6.

4.2.2.2 Normalização da função semelhança

Conforme sugere (NUMMIARO; et al.,2003), após obter-se o valor de semelhança entre a amostra e o modelo armazenado da bola, calcula-se a distância entre ambos utilizando (26).

$$\delta = \sqrt{1 - sem(im_{templ}, im)} \tag{26}$$

Após, utiliza-se uma função gaussiana (27) normalizada entre [0;1] e com variância σ para normalizar o valor de δ. Dessa forma dá-se ênfase aos valores mais parecidos com o modelo, que possuem distâncias menores. Essas amostras possuem maiores chances de serem escolhidos enquanto outras amostras com distâncias maiores serão naturalmente descartadas do processo.

$$\pi^{(n)} = p\left(z_t \middle| x_t = s_t^{(n)}\right) = e^{-\frac{\delta^2}{2\sigma^2}} \tag{27}$$

4.2.2.3 Determinação automática de $\boldsymbol{\sigma}$

A escolha de σ muito grande acarretará em pesos exagerados para pontos que não são realmente a bola. Já um σ muito pequeno acarretará que mesmo amostras que coincidem com a bola a possuiam um valor de $\pi^{(n)}$ muito pequeno. Mesmo assim, (NUMMIARO; et al.,2003) sugere a escolha manual para o valor de σ.

Para contornar esse problema é proposto nesse trabalho um cálculo automático para determinar σ. Esse cálculo estima o valor médio (μ_a) e de variância (σ_a^2) do valor δ, calculado em (26), de N amostras aleatórias na imagem. Esse valor médio e de variância é considerado como o valor médio e a variância da semelhança do fundo com a bola. (ISARD; et al., 1998) e (NUMMIARO; et al.,2003) sugerem o número de amostras entre 100 e 1000. Nos experimentos, N foi fixado em 200, por apresentar melhores resultados sem impactar drasticamente no desempenho computacional.

Como esperamos que, na média, o fundo possua uma semelhança baixa com a bola, σ é então calculado de maneira que $\mu_a + \sigma_a^2$ possua um valor de $\pi^{(n)}$ baixo, conforme mostra (28). O valor de $sem(im_{templ}, im) = \mu_a + \sigma_a^2$ foi escolhido, pois representam 85% das amostras da imagem, se considerar a distribuição de semelhanças como sendo gaussiana.

$$\sigma^2 = \frac{(1 - \mu_a - \sigma_a^2)}{2\ln(L_a)} \tag{28}$$

Desse modo, cada vez que o rastreador é inicializado, σ é calculado. Os resultados dessa solução são discutidos na seção 6.

5 IMPLEMENTAÇÃO

O programa foi implementado em C++ utilizando uma divisão por classes, conforme indica a Figura 22.

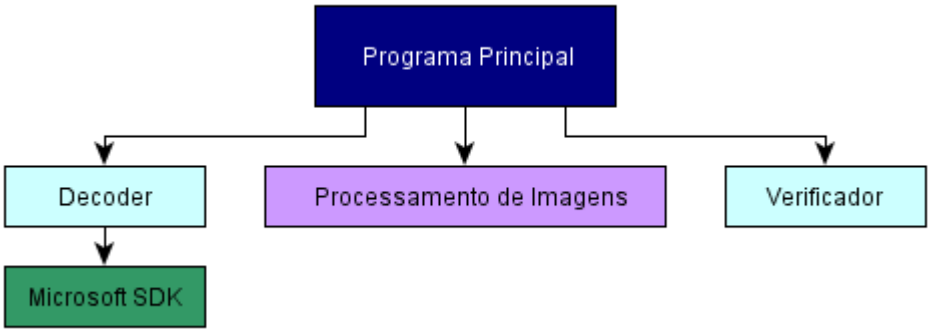


Figura 22 – Divisão do Programa Principal

Para decodificar o vídeo foi utilizado o decodificador DirectShow, um dos componentes do DirectX. Essa API está incluída na Microsoft SDK, de distribuição gratuita. Após a decodificação os frames foram armazenados em um buffer para serem processados em ordem.

O processamento de imagens foi baseado na biblioteca Lili2. Esta foi desenvolvida no LaPSI e é gratuita e de uso livre para fins não comerciais. A divisão

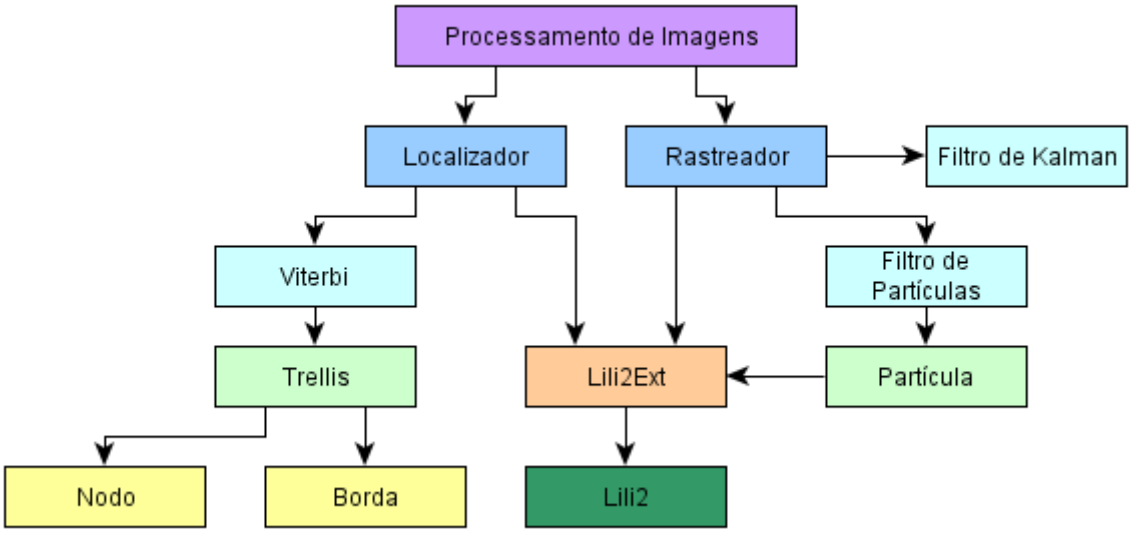


Figura 23 – Classes da etapa de Processamento de Imagens

de classes e suas relações podem ser vistas na Figura 23.

Sobre essa biblioteca foi criada uma classe LiliExt que continha as funções de processamento bruto de imagens, como, por exemplo, segmentação, crescimento de regiões por agrupamento de pixels, entre outras.

Para as outras etapas de processamento, foram criadas classes específicas. Foram criadas, ainda, classes de monitoramento e de diagnóstico a fim de extrair automaticamente as informações de performance e de avaliação de acertos.

Uma interface com o usuário foi desenvolvida para facilitar a utilização do programa. Conforme pode ser vista na Figura 24, ela possui duas telas de visualização. Na da esquerda é mostrado o vídeo com o(s) candidato(s) marcados. E aquele que foi escolhido como bola marcado em destaque (vermelho). Ela possui, também, os botões de comando de vídeo – play, pause e step-frame. Foi inserida uma opção para salvar as imagens geradas pelo algoritmo e outra opção para

Figura 24 – Interface com o usuário

marcar não apenas o candidato escolhido como bola e sim todos os candidatos encontrados e não descartados na etapa de eliminação. A interface permitia o usuário carregar um vídeo do tipo AVI, uma sequência de imagens bitmap ou ainda o último vídeo decodificado, que ficava armazenado no computador.

Toda a configuração do algoritmo, como constantes, seleção do algoritmo de rastreamento e da função de similaridade é feito pelo menu configurar.

A implementação foi desenvolvida, ainda, em um ambiente multi-threads, para obter o maior desempenho das arquiteturas de processadores modernos.

6 TESTES E RESULTADOS

Conforme mostrado nos capítulos anteriores, algumas alternativas foram propostas para resolver o problema de localizar e rastrear a bola. Rastreamento utilizando Filtro de Kalman ou Filtro de Partículas foram discutidos, bem como a influência da escolha da função de semelhança na performance do Filtro de Partículas.

Dessa forma foram gerados seis cenários para avaliar qual dessas escolhas é mais indicada para cada caso. O primeiro cenário, chamado Cenário 1, que servirá de referência para comparação dos outros, constitui apenas do localizador. Ele nunca passa para a etapa de rastreamento. O segundo cenário, chamado Cenário 2, o rastreador é consistido por um filtro de Kalman. Os últimos quatro cenários correspondem ao Filtro de Partículas com quatro possibilidades de função de semelhança: Semelhança de Histograma (Cenário 3), Correlação Cruzada em Tons de Cinza (Cenário 4), Correlação Cruzada da Imagem Segmentada (Cenário 5) e Semelhança de Projeção nos Eixos dos Pontos Segmentados (Cenário 6).

Para avaliar os cenários descritos, as demais constantes do algoritmo foram fixadas em valores idênticos para todos os cenários, conforme os valores acima sugeridos.

Foram selecionados 21 vídeos para realizar os testes. Na sua maioria, os vídeos utilizados foram retirados de trechos do jogo Brasil x Japão da Copa de 2006, gentilmente cedido pela FIFA®. Os demais vídeos testados foram obtidos de jogos diversos e selecionados conforme sua qualidade. Os vídeos foram divididos conforme as características relevantes que podem alterar o desempenho do algoritmo. Conforme visto no Capítulo 1, linearidade de movimento, mudanças de direção e velocidade, oclusões, presença de público e distância da câmera são

características que podem afetar o localizador e o rastreador e foram consideradas na divisão dos vídeos, conforme pode ser visto na Tabela 1.

TABELA 1 – TIPOS DE VÍDEOS

Característica	Tipo I	Tipo II	Tipo III	Tipo IV	Tipo V
Oclusão	Não	Parcial	Curta	Curta	Longa
Movimento da Bola	Linear	Linear	Truncado	Parabólico	Truncado
Borramento	Ausente	Baixo	Baixo	Alto	Severo
Velocidade da Bola	Baixa	Baixa	Alta	Alta	Alta
Platéia	Ausente	Presente	Presente	Presente	Presente

A Tabela 2 apresenta as características básicas dos vídeos utilizados. O número de frames médio é 169. A freqüência utilizada foi 25 frames por segundo para todos os vídeos, dessa forma obtém-se uma média de 6s76 por vídeo.

TABELA 2 – RELAÇÃO DOS VÍDEOS UTILIZADOS

Vídeo	Tipo	Resolução	N° frames	Tempo(s)
Vídeo 1	I	768x576	58	2s32
Vídeo 2	I	720x576	118	4s72
Vídeo 3	II	720x576	149	5s96
Vídeo 4	II	768x576	196	7s84
Vídeo 5	II	768x576	91	3s64
Vídeo 6	II	720x576	71	2s84
Vídeo 7	II	720x576	136	5s44
Vídeo 8	III	720x576	318	12s72
Vídeo 9	III	768x576	110	4s40
Vídeo 10	III	720x576	237	9s48
Vídeo 11	III	720x576	341	13s64
Vídeo 12	III	320x240	175	7s00
Vídeo 13	IV	768x576	96	3s84
Vídeo 14	IV	720x576	245	9s80
Vídeo 15	IV	720x576	156	6s24
Vídeo 16	IV	720x576	173	6s92
Vídeo 17	V	720x576	93	3s72
Vídeo 18	V	720x576	292	11s68
Vídeo 19	V	720x576	171	6s84
Vídeo 20	V	720x576	121	4s84
Vídeo 21	V	768x576	205	8s20

Em cada um dos vídeos, para cada frame, foi realizado um procedimento de marcação manual da posição correta da bola. Dessa marcação obteve-se um padrão que foi armazenado em um arquivo, permitindo sua utilização para comparação posterior. Dessa forma definimos que, em cada frame, a bola é considerada encontrada pelo algoritmo caso a posição prevista esteja dentro de um

quadrado ao redor da posição do modelo. O quadrado possui um tamanho configurável e para esse trabalho ele foi definido como sendo de 20 pixels de lado.

Assim, o desempenho do algoritmo pode ser medido dividindo-se o número de frames onde a bola foi encontrada pelo número total de frames do vídeo. As tabelas a seguir mostram o desempenho dos cenários para os vídeos divididos em seus respectivos tipos.

Nos vídeos de Tipo I espera-se que o algoritmo obtenha a máximo desempenho. Os resultados dos testes podem ser vistos na Tabela 3. Como se pode observar, todos os cenários obtiveram resultados excelentes em ambos os vídeos.

TABELA 3 – DESEMPENHO DOS CENÁRIOS NOS VÍDEOS DE TIPO I

	Vídeo 1	Vídeo 2
Cenário 1	97%	76%
Cenário 2	97%	86%
Cenário 3	86%	84%
Cenário 4	98%	93%
Cenário 5	95%	88%
Cenário 6	100%	91%

Nesses vídeos podemos destacar o Cenário 6 que obteve 100% de aproveitamento no Vídeo 1 e 91% no Vídeo 2. Outro cenário que obteve resultados excelentes foi o Cenário 4, que também obteve valores sempre superiores à 90% em ambos os vídeos.

Nos vídeos de Tipo II, cujos resultados são mostrados na Tabela 4, podem-se ver com clareza as diferenças entre os diversos tipos de rastreadores, uma vez que se podem separar aqueles que ampliaram o desempenho do localizador, daqueles que não apresentaram melhoras significativas.

As diferenças de desempenho, quando comparados aos resultados obtidos com os vídeos do Tipo I, podem ser explicadas pela presença de platéia (aumento

de ruído no ambiente) e presença de oclusões parciais. Essas oclusões representam uma perda significativa no desempenho dos cenários 1 e 2.

TABELA 4 – DESEMPENHO DOS CENÁRIOS NOS VÍDEOS DE TIPO II

	Vídeo 3	Vídeo 4	Vídeo 5	Vídeo 6	Vídeo 7
Cenário 1	53%	73%	60%	58%	49%
Cenário 2	93%	87%	62%	75%	68%
Cenário 3	80%	62%	77%	63%	47%
Cenário 4	86%	84%	87%	71%	75%
Cenário 5	83%	73%	77%	75%	71%
Cenário 6	82%	69%	81%	75%	73%

Como visto na tabela acima, os rastreadores dos cenários 1 e 2 tiveram uma perda de performance significativa, enquanto os demais, que utilizam o Filtro de Partículas, mantiveram a taxa de acertos alta, mesmo com a presença de ruído, oclusões parciais e a presença, ainda que pequena, de borramento. A exceção foi apresentada pelo Cenário 3, que deve desempenho inferior em alguns vídeos. Esse desempenho pode ser explicado pela própria definição do filtro, que rastreia objetos com histograma semelhante àquele apresentado pelo modelo obtido na etapa de localização (verde e branco, principalmente). Se a bola for ocluída por um objeto que possuir histograma semelhante ao seu, o filtro será “atraído” para este, ocasionando erro no rastreamento do objeto verdadeiro, no caso, a bola.

Os Cenários 4, 5 e 6 obtiveram resultados muito semelhantes e satisfatórios para o tipo de vídeo. O destaque, novamente, ficou com o Cenário 4, que sempre ficou entre os 2 melhores rastreadores em todos os vídeos, com exceção ao Vídeo 6, onde ainda assim teve 71% de acerto, contra 75% do melhor rastreador naquele vídeo. Entretanto, apenas com esses dados, não pode-se afirmar que nenhum dos rastreadores é efetivamente melhor que os outros.

As diferenças significativas entre os cenários que utilizam o filtro de partículas começam a aparecer nos vídeos do Tipo III, conforme é mostrado na Tabela 5.

TABELA 5 – DESEMPENHO DOS CENÁRIOS NOS VÍDEOS DE TIPO III

	Vídeo 8	Vídeo 9	Vídeo 10	Vídeo 11	Vídeo 12
Cenário 1	70%	45%	61%	55%	44%
Cenário 2	77%	43%	58%	57%	62%
Cenário 3	61%	72%	61%	45%	72%
Cenário 4	68%	61%	61%	60%	69%
Cenário 5	33%	59%	39%	32%	62%
Cenário 6	62%	49%	60%	54%	72%

Nessa tabela, pode-se notar que todos os Cenários começam a apresentar desempenhos inferiores àqueles apresentados nos vídeos do Tipo II, devido às condições de borramento e oclusões mais severas apresentadas pelos vídeos. Porém, nota-se que o Cenário 5 apresenta um desempenho consideravelmente abaixo dos outros. Isso se deve ao fato de que a função de cálculo de semelhança utilizado nesse cenário não lidar bem com o alto borramento da bola, nem com oclusões mais severas presentes nos vídeos do Tipo III.

Novamente, podemos ressaltar que o Cenário 4 obteve desempenho superior aos demais, com uma média alta e sem nenhum desempenho insatisfatório, ficou sempre entre os três melhores de cada vídeo, sendo que nos Vídeos 10 e 11 o que obteve melhor resultado. Foi, também, o único a ter sempre média de acerto superior a 60%.

Na Tabela 6, podemos ver os resultados dos cenários aos vídeos de Tipo IV. Nesse tipo de vídeo, espera-se um desempenho ainda inferior dos Cenários. Entretanto, no Vídeo 13, todos os cenários apresentaram média superior aos demais vídeos do Tipo IV e até superior aos vídeos de outros tipos considerados mais fáceis de rastrear. Isso se deve ao fato de que o vídeo, apesar de possuir borramento muito forte – que o classifica como Tipo IV –, foi filmado com um zoom muito próximo. Assim, mesmo que o rastreador perca a bola, ela pode ser encontrada novamente pelo localizador, mesmo que borrada, uma vez que apresenta muitos pixels para representá-la.

TABELA 6 – DESEMPENHO DOS CENÁRIOS NOS VÍDEOS DE TIPO IV

	Vídeo 13	Vídeo 14	Vídeo 15	Vídeo 16
Cenário 1	68%	48%	56%	47%
Cenário 2	75%	50%	47%	49%
Cenário 3	78%	51%	37%	39%
Cenário 4	76%	53%	59%	48%
Cenário 5	73%	42%	29%	20%
Cenário 6	77%	48%	61%	52%

Em face disso, levando em conta os demais vídeos, nota-se que o Cenário 5 apresentou, novamente, resultados muito abaixo dos resultados apresentados pelos demais. Nesse caso podemos destacar os Cenários 4 e 6, que obtiveram os melhores resultados, foram os únicos que, em todos os vídeos do tipo IV, nunca apresentaram desempenho inferior ao Cenário 1 – que possui apenas o localizador.

Por fim, nos vídeos do Tipo V, o desempenho de todos os Cenários foi muito inferior aos mostrados nos vídeos dos outros tipos. Os dados podem ser vistos na Tabela 7. Isso pode ser considerado um indício de que o algoritmo deve ser estudado e aperfeiçoado para os casos onde as condições do vídeo são as mais severas possíveis.

TABELA 7 – DESEMPENHO DOS CENÁRIOS NOS VÍDEOS DE TIPO V

	Vídeo 17	Vídeo 18	Vídeo 19	Vídeo 20	Vídeo 21
Cenário 1	38%	28%	40%	37%	48%
Cenário 2	38%	37%	51%	29%	49%
Cenário 3	38%	34%	27%	34%	41%
Cenário 4	34%	38%	48%	46%	42%
Cenário 5	29%	31%	34%	47%	12%
Cenário 6	34%	37%	32%	47%	44%

7 CONCLUSÕES

Analisando os resultados obtidos e discutidos no capítulo anterior, podemos destacar alguns pontos importantes.

A utilização de apenas o algoritmo de localização, sem utilizarmos um rastreador, acima representado pelo Cenário 1, não apresenta resultado satisfatório, ficando em muitos dos casos com desempenho muito aquém daquele mostrado pelos outros cenários. Esse desempenho se explica pelo fato de que, em um jogo real de futebol, como em qualquer cenário de rastreamento de objetos em ambientes ruidosos, o objeto de interesse está sujeito a oclusões e interferências dos outros objetos que compõem a cena. Ficou claro também que o desempenho do localizador interfere na qualidade do rastreador, uma vez que os dados de entrada do mesmo serão obtidos do primeiro, que, ao fornecer dados insuficientemente precisos comprometerão toda a etapa de rastreamento.

A proposta de utilizar a projeção dos pontos segmentados como um classificador para o filtro de partículas se mostrou ineficaz, como podemos observar no desempenho do Cenário 5, que continha a sua implementação. Isso se deve ao fato de a função ser muito pouco específica ao comparar o ponto onde o rastreador sugeriu a bola e o modelo obtido do localizador. Assim, em muitos casos, o rastreador acabava ficando preso em regiões que, embora visualmente não se parecessem com a bola, possuíam valor no classificador suficiente para que não fossem descartados completamente.

O filtro de partículas baseado em histograma a cores, proposto por (NUMMIARO; et al.,2003) e implementado no cenário 3, obteve desempenho insatisfatório. Isso pode ser atribuído ao fato de a bola possuir um histograma “pobre”, sendo formado basicamente por duas cores: a cor da bola – aqui

considerada branca – e a cor do fundo – normalmente verde, devido à presença do gramado. Assim qualquer objeto que ocluir parcialmente ou totalmente a bola, e possuir histograma semelhante (meia, chuteira, linha ou outro objeto branco), irá interferir no rastreamento da mesma.

A utilização do filtro de Kalman apresentada por (LIU; et. al., 2006), aqui representada pelo Cenário 2, mostrou-se limitada, uma vez que apresentou resultado inferior aos Cenários 4 e 6 na maioria dos casos. Os valores de acerto se mostraram satisfatórios em situações onde não havia oclusões parciais ou totais, pois a maneira de verificar o resultado do filtro de Kalman, e consequentemente atualizá-lo, considera que será encontrado um objeto circular (a bola) na região onde foi previsto a localização da bola, fato que não ocorria quando havia oclusões parciais presentes no vídeo.

Os Cenários 4 e 6 foram os que apresentaram os melhores resultados. O uso da correlação cruzada para determinar a semelhança com o modelo apresentou uma forma robusta além de ser flexível, pois podia diferenciar, com valores de semelhança intermediários, a bola mesmo quando esta estivesse parcialmente oculta. Outra grande vantagem desses métodos é que apresentaram valores muito diferentes para objetos que se aproximassem da bola sem interferir na medida da mesma.

A diferença entre o método 4 e 6 é que, por definição, a correlação cruzada em escala de tons de cinza é mais precisa que a correlação binária, pois apresenta uma maior granularidade no seu resultado. Assim, o Cenário 4 apresentou resultados ligeiramente melhores que o 6, o que pode ser visto nos resultados apresentados no capítulo anterior.

Todos os cenários apresentaram tempos de execução semelhantes. Para os vídeos de tamanho maior (720x576) não se pode considerar a ideia de utilizar os algoritmos em tempo real. O tempo de processamento ficou aproximadamente o dobro do tempo do vídeo. Reduzindo-se a resolução do vídeo ou modificando a quantidade de frames por segundo pode contornar esse problema.

Como trabalhos futuros, existem diversas possibilidades. No campo das aplicações esportivas, o algoritmo poderia incorporar a detecção do gramado e dos jogadores, e assim aumentar a robustez do rastreador, uma vez que a bola não poderia estar fora do campo por tempo prolongado nem sem se movimentar. Poderia eliminar também os casos em que se encontra a bola como sendo uma parte de um jogador (calção, meia, etc).

No campo das aplicações de rastreamento em geral, pode-se desenvolver a modelagem de outros objetos de interesse para serem localizados e rastreados, uma vez que, para localizar e rastrear outros objetos, basta que se modifique suas características no cálculo dos pesos de entrada do algoritmo de viterbi e, no caso do Cenário 2, se altere a função de semelhança para o novo modelo.

Com tais mudanças, o algoritmo se credencia para uma nova gama de aplicações, como por exemplo, vigilância, controle de tráfego, sistemas militares de defesa e ataque, localização e rastreamento de outros objetos de interesse em diversos esportes.

REFERÊNCIAS

BALLARD, Dana H.; BROWN, Christopher M. *Computer Vision*. Englewood Cliffs : Prendice-Hall, 1982. 0-13-165316-4.

CHOI, Kyuhyoung; SEO, Yongdeuk. Probabilistic tracking of the soccer ball. *Statistical Methods in Video Processing*. Berlin : Springer Berlin / Heidelberg, 2004, v. 3247, p. 50-60.

EBNER, Marc. 2007. *Color Constancy*. Nova Iorque : Wiley, p. 408. 978-0-470-05829-9.

FÉDÉRATION INTERNATIONALE DE FOOTBALL ASSOCIATION (FIFA). *2006 FIFA World Cup(tm) Japan V Brazil - Whole Match*. DVD cedido sob contrato n° AS2098.

FORNEY, George David Jr. The Viterbi Algorithm. *IEEE Procedings*. 1973,v. 61, n. 3, p. 268-278.

GONZALEZ, Rafael C.; WOODS, Richard E. *Processamento de Imagens Digitais*. São Paulo : Editora Blücher, 2000. 85-212-0264-4. Tradução de Roberto Marcondes Cesar Jr. e Luciano da Fontoura Costa

ISARD, Michael; BLAKE, Andrew. Condesation - conditional density propagation for visual tracking. *International Journal of Computer Vision*. 29, n. 1, p. 5-28, 1998.

ISHII, Norihiro, et al. 3D tracking of a soccer ball using two synchronized cameras. *Advances in Multimedia Information Processing – PCM*. Heidelberg. v. 4810, 2007.

JAIN, Anil K. *Fundamentals of Digital Image Processing*. Englewood Cliffs: Prentice-Hall, 1989. 0-13-336165-9.

KIM, Taeone; SEO, Yongduek; HONG, Ki-Sang. Physics-based 3D position analysis of a soccer ball from monocular image sequences. In: INTERNATIONAL CONFERENCE ON COMPUTER VISION, 6, Washington DC: Anais IEEE Computer Science, p. 721-726, 1998.

LIU, Yang, et al. Extracting 3D information from broadcast soccer video. *Image and Vision Computing*. v. 24, p. 1146–1162, Apr 19, 2006.

NUMMIARO, Katja; KOLLER-MEIER, Esther; VAN GOOL, Luc. An adaptive color-based particle filter. *Image and Vision Computing*. v. 21, p. 99-110, 2003.

OK, Hyun-Wook; SEO, Yonduek; HONG, Ki-Sang. Multiple soccer players tracking by condensation with occlusion alarm probability. In: INTERNATIONAL WORKSHOP ON STATISTICAL METHODS FOR VISION PROCESSING, *Procedings..* Copenhagen, Denmark. IEEE Transactions, 2002.

PIOVOSO, Michael; LAPLANTE, Phillip A. Kalman filter recipes for real-time image processing. *Real-Time Imaging*. v. 9, p. 433-439, 2003.

RABAEY, Jan M.; CHANDRAKASAN, Anantha; NIKOLIĆ, Borivoje. Design project: designing a viterbi decoder. *Digital Integrated Circuits*. 2ª Ed. s.l. : Prentice Hall, 6 e 11.

SEO, Yongduek, et al. Where are the ball and players? soccer game analysis with color-based tracking and image mosaick. *Image Analysis and Processing*. Berlin : Springer Berlin / Heidelberg v. 1311, p. 196-203, 1997.

TANG, Feng; TAO, Hai. Object tracking with dynamic feature graph. *Visual Surveillance and Performance Evaluation of Tracking and Surveillance*. p. 25-32, 16 Oct 2005

WELCH, Greg; BISHOP, Gary. An Introduction to the kalman filter. [Online] 24 July 2006. Disponível em: <*http://www.cs.unc.edu/~welch/kalman/ kalmanIntro.html*> Acesso em: 01 de Dezembro de 2008.

XU, M., et al. Architecture and algorithms for tracking football players with multiple cameras. *Vision, Image and Signal Processing, IEE Proceedings*. v. 152, n. 2, p. 232-241. 8 Apr 2005

YU, Xinguo, et al. Trajectory-based ball detection and tracking with applications to semantic analysis of broadcast soccer video. In: *ACM INTERNATIONAL CONFERENCE ON MULTIMEDIA*. 2003, Berkeley, USA, *ACM Proceedings*, New York, ACM, p. 11-20, 2003.

YU, Xinguo; TIAN, Qi; WAN, Kong Wah. A novel ball detection framework for real soccer video. In: INTERNATIONAL CONFERENCE IN MULTIMEDIA AND EXPO, 2, 2003. Baltimore. *IEEE Proceedings…* Nova Iorque: IEEE Computer Science. p. 273-6, 2003.

ANEXO: A NEW COLOR SPACE FOR OPTIMIZING CHROMATICITY DISTANCE BASED ALGORITHMS

ANEXO: A NEW COLOR SPACE FOR OPTIMIZING CHROMATICITY DISTANCE BASED ALGORITHMS

Alberto do Canto, André B. Soares, Letícia V. Guimarães, Marcelo Negreiros, Vinícius Souza and Altamiro A. Susin

Electrical Engineering Department - Federal University of Rio Grande do Sul - UFRGS
Porto Alegre, Brazil

ABSTRACT

This work presents a new color space able to optimize chromaticity distance based algorithms like edge detection and segmentation. Distance is measured using rectangular coordinates in the chromaticity plane. This method results in a reduced number of operation in comparison to methods like vector angle in RGB color space and HS plane. The technique is compared to other current approaches to evaluate chromaticity distance. Results show a performance gain of up to four times against vector angle and 42% against normalized RGB in 3x3 neighborhood operations.

Index Terms— Color space, chromaticity distance

1. INTRODUCTION

Color information is essential for many image processing applications. Intensity variations, however, must be disregarded in some cases, for example, when resulted from heterogeneous illumination. Chromaticity distance measurements for pixels in images represented in the RGB color space have a large influence of intensity values, even though the pixel color is the same.

Algorithms based on color distance like edge detection and segmentation [7] can benefit from the proposed approach. In figure 1 it is shown an example where intensity only images (figure 1b) do not enable the detection of zones of interest. Combined color only (figure 1c) and saturation only (figure 1d) images enable the differentiation of cells from other image artifacts.

Although other color spaces made it possible to isolate the intensity from color (as the HSI color space) [5], the computational load of converting an image to another space and getting it back to RGB can be too high. Furthermore, combined color and saturation effects in pixel distance evaluations in the HSI color space may be difficult to obtain because of additional mathematical operations needed.

In this work it is developed an efficient method to perform pixel distance evaluation considering color and saturation, while minimizing the influence of intensity. A new color space is presented, the WXY color space, that has the following characteristics: low computational load for converting to and from RGB color space; separates intensity from other pixel characteristics and has small processing requirements for chromaticity distance evaluation.

This paper is organized as follows. Section 2 presents a brief review of previous work in chromaticity distance measurements in color images. The WXY color space is presented in section 3. Section 4 presents comparison results of the method against other current approaches. Analysis is presented in section 5, followed by conclusions and further work in section 6.

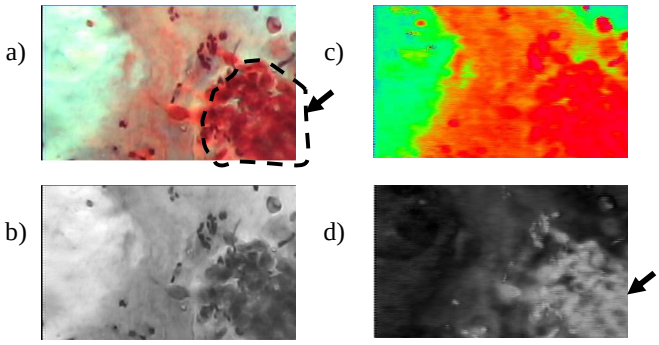


Figure 1. The edge detection problem (ROI Identification) in cytological images: (a) original image and region of interest (cell group); (b) intensity representation, difficult to guess the ROI; (c) color information and (d) saturation information enable easier ROI differentiation from other image artifacts.

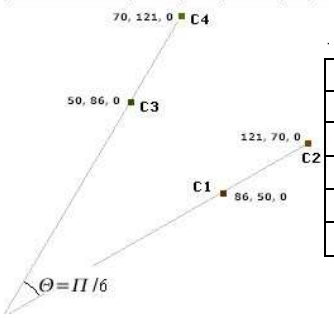
2. PREVIOUS WORK

The problem of combined color and saturation effects in pixel distance evaluation has already been addressed in the literature [1-3]. While the RGB color space itself does not provide any resource for disregarding intensity, the angular distance between two pixels in the RGB space can be used as a distance measurement in order to disregard intensity values in the RGB color space [1-3].

Given two pixels, P1=(R_1,G_1,B_1) and P2=(R_2,G_2,B_2), the distance measured as suggested in [1] using the sine of the vector angle is

$$d(P1,P2) = \sin(\Theta) = \sqrt{1 - \frac{(R_1R_2 + G_1G_2 + B_1B_2)^2}{(R_1^2 + G_1^2 + B_1^2)\cdot(R_2^2 + G_2^2 + B_2^2)}} \quad (1)$$

In figure 2, C1 and C2 (also C3 and C4) are collinear vectors with identical color but different intensities. The chromaticity distance evaluated using (1) is presented in figure 2.



d(C1,C2)	0
d(C3,C4)	0
d(C1,C3)	sin(π/6)
d(C1,C4)	sin(π/6)
d(C2,C3)	sin(π/6)
d(C2,C4)	sin(π/6)

Figure 2. Vector angle distance measurements.

However, as presented by equation (1), the computational load associated is high. The normalized RGB color space is able to disregard intensity in the RGB space [6].

The HSL (also called HLS or HSI) color space [4,5] provides resources for separating intensity from color. However, situations where combined saturation and color effects are taken into account have additional computational load, since evaluating distances in the HS plan involves evaluating distances in polar coordinates. Furthermore, the computational load for converting from RGB to HSL is high.

In this work a color space similar to the HSI is developed using Cartesian coordinates, allowing chromaticity distance operations with low computational costs. The WXY color space is presented in the next section.

3. THE WXY COLOR SPACE

Consider the vector C expressed as a linear combination of the base vectors in the RGB space and the primary colors Cr, Cg, Cb:

$$\vec{C} = C_r\vec{r} + C_g\vec{g} + C_b\vec{b}\,, \quad (2)$$

where $\vec{r} = (1,0,0), \vec{g} = (0,1,0), \vec{b} = (0,0,1)$.

The vector C can be expressed in the proposed WXY color space as

$$\vec{C} = C_w'\,\vec{w} + C_x'\,\vec{x} + C_y'\,\vec{y} \quad (3)$$

where

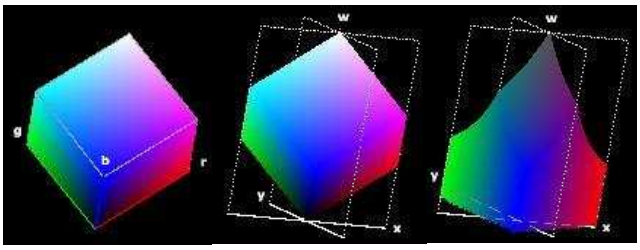
$$\vec{w} = (1/3,1/3,1/3)$$
$$\vec{x} = (1/6,-1/12,-1/12) \quad (4)$$
$$\vec{y} = (0,1/12,-1/12).$$

The transformation from RGB to WXY can be performed by

$$[C_r\ \ C_g\ \ C_b] = [C_w'\ \ C_x'\ \ C_y']\cdot\begin{bmatrix}4 & 4 & 4\\ 2 & -1 & -1\\ 0 & 1 & -1\end{bmatrix}\cdot\frac{1}{12} \quad (5)$$

$$[C_w'\ \ C_x'\ \ C_y'] = [C_r\ \ C_g\ \ C_b]\cdot\begin{bmatrix}1 & 4 & 0\\ 1 & -2 & 6\\ 1 & -2 & -6\end{bmatrix} \quad (6)$$

After inspecting (5) and (6) it becomes clear that the transformation process is not very demanding, since only additions and multiplications are used. In figure 3, it is shown the RGB color space (figure 2a) and the proposed WXY color space (figure 3b). In the WXY space, the *W* axis represent intensity and the *X* and *Y* axis correspond to the pixel chromaticity.



(a) (b) (c)

Figure 3. Pixel mapping: (a) in the RGB color space, (b) in the WXY color space, and (c) in the normalized WXY color space.

In the transformation presented in (6), the components C_x' and C_y' will be influenced by intensity, even for pixels with the same color. This is accounted for in the normalized WXY color space, as presented in figure 3c and defined by equation 7.

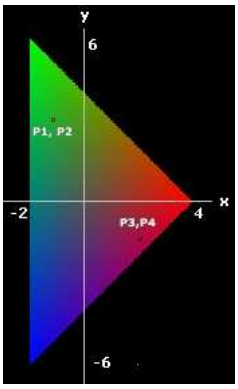
$$C_w = C_w'$$
$$C_x = C_x'/C_w' \quad (7)$$
$$C_y = C_y'/C_w'$$

The space represented in figure 3b is an intermediary step used only in the color space conversion. The normalized WXY color space will be used in the remainder of this paper.

One should observe that, as the base of the WXY color space is not canonical (the base vectors have different module), the distance equation must be compensated. This fact has been taken into account in the chromaticity distance equation (8). Considering two pixels represented in the WXY color space, P1 = (w1, x1, y1) and P2 = (w2, x2, y2), the chromaticity distance between then is given by

$$d = \sqrt{3(x1-x2)^2 + (y1-y2)^2}\,. \quad (8)$$

The plane of chromaticity is presented in figure 4. Four pixels are shown, and their coordinates in the RGB and WXY spaces are presented in table 1. Pixels P1 and P2 (and P3 and P4) have the same color but different intensities. Distances between these pixels are presented in figure 4.



d(P1,P2)=0
d(P1,P3)=7.21
d(P1,P4)=7.21
d(P2,P3)=7.21
d(P2,P4)=7.21
d(P3,P4)=0

Figure 4. Plane of chromaticity.

Table 1. Pixels in figure 4 (RGB and WXY space)

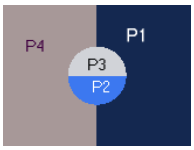
	RGB	WXY
P1	(40,190,50)	(280,-1.142,3)
P2	(20,95,25)	(140,-1.142,3)
P3	(200,10,80)	(290, 2.137,-1.448)
P4	(300,15,120)	(435,2.137,-1.448)

4. EXPERIMENTAL RESULTS

In this section a qualitative comparison of chromaticity distance measurements using different color spaces is performed. After the proposed method is applied to an edge detection algorithm.

4.1. Qualitative comparison

An edge detection algorithm (Roberts [4,5]) was applied to the test image in figure 5a. The points P1 to P4 are presented in table 2.

a) 

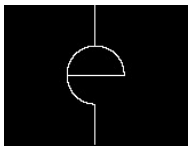
b) 

Figure 5. a)Test image and b) chromaticity edge detection template.

Table 2. Pixels in test image for different color spaces

	RGB	WXY	HSI	RnGnBn
P1	(20,40,80)	(47,-1.14,-1.71)	(147,144,47)	(36,73,146)
P2	(60,120,240)	(140,-1.14,-1.71)	(147,206,141)	(36,73,146)
P3	(209,211,216)	(212,-0.03,-0.05)	(147,20,200)	(84,85,87)
P4	(167,152,152)	(157,0.13,0.00)	(239,20,150)	(90,82,82)

Table 3. Feature similarity for pixels in figure 5a

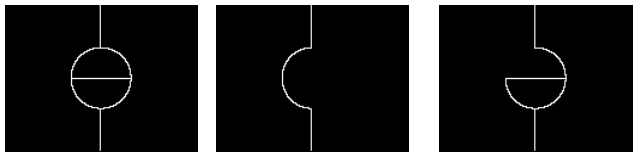
Feature	similar pixels
Chromaticity	P1,P2
Hue	P1,P2,P3
Saturation	P3,P4

Table 3 shows that only pixels P1 and P2 have equal chromaticity, although intensities are different. This defines figure 5b as a chromaticity edge detection template.

In figure 6a the edges detected in the RGB color space using Euclidean distance is presented. The result confirms that the RGB space is not adequate to measure chromaticity distances [2].

Edges detected in the HSI space using hue distance are presented in figure 6b, and using saturation distance are presented in figure 6c. In the HSI space the differences in chromaticity will influence both hue and saturation, so this color space is also not adequate to measure chromaticity distances. Although both hue and saturation could be combined, the associated computational overhead would be high.

The normalized RGB color system (RnGnBn)[6], the WXY color system and vector angle measurement in the RGB color system were also used in this evaluation and achieved similar results to the template shown in figure 5b.



a b c

Figure 6. Edges detected by: a) RGB and Euclidean distance; b) HSI and hue distance and c) HSI and saturation distance.

4.2. Performance comparison

The algorithms that have successfully detected the template in the previous subsection were used in a performance comparison test. First, images are taken in RGB format. After that, chromaticity distance is evaluated directly in RGB space for the vector angle algorithm, or after space conversion, for the normalized RGB and WXY color spaces. Color space conversion and distance evaluation times were measured separately. Results are presented in table 4. Quadratic distances and integer arithmetic were used to enhance the performance of the algorithms. Time results were normalized to the vector angle algorithm.

Table 4. Performance comparison for a single chromaticity distance evaluation

Operation	Normalized Time (%)
RGB to WXY conversion: $W' = R + G + B$ $X = 512(2R - G - B)/W'$ $Y = 1536(G - B)/W'$	107
Distance (WXY space): $d^2 = 3(x1 - x2)^2 + (y1 - y2)^2$	10
RGB to normalized RGB conversion: $I = R + G + B$ $r = 256 * R / I$ $g = 256 * G / I$ $b = 256 * B / I$	142
Distance (normalized RGB): $d^2 = (r1 - r2)^2 + (g1 - g2)^2 + (b1 - b2)^2$	16
Distance (Vector Angle): $d^2 = 1 - \frac{(R_1R_2 + G_1G_2 + B_1B_2)^2}{(R_1^2 + G_1^2 + B_1^2)\cdot(R_2^2 + G_2^2 + B_2^2)}$	100

5. ANALYSIS

Table 4 shows that the WXY color space has a better performance than the normalized RGB color space, considering both space conversion and distance evaluation.

Distance evaluation using vector angle method in the RGB color space does not have the cost of color space conversion, so it is better for an algorithm that uses only one chromaticity distance evaluation per pixel.

Nevertheless, for algorithms that use more than one chromaticity distance evaluation per pixel, the method based on the WXY color space presents a better performance. This happens because the chromaticity distance evaluation has a lower cost in this space. This is clearly shown in figure 7, where up to eight comparisons are performed considering a 3x3 pixel's neighborhood.

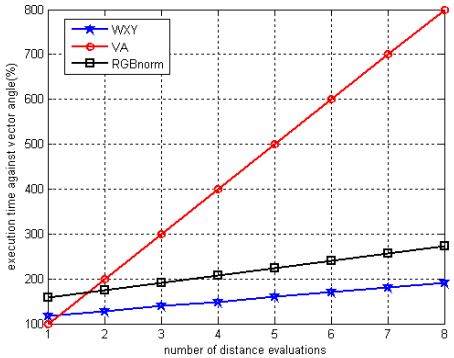


Figure 7.Performance comparison for different number of distance evaluations between normalized RGB, VA and WXY.

6. CONCLUSIONS

A high performance chromaticity distance based on a new color space was presented and characterized in this paper. Results show a performance gain up to four times compared with current approaches. The advantages of using a color space that allows a high performance chromaticity distance evaluation can be noticed in applications like edge detection, filtering and segmentation.

Figure 8 presents a segmentation example that was performed using chromaticity distance. The region of interest shows a cell group. This application (automating screening in pap smears exams) is an example where performance is critical because of the large number of images and chromaticity information that need to be analyzed.

Any algorithm based on chromaticity distance evaluations can benefit from using the WXY color space.

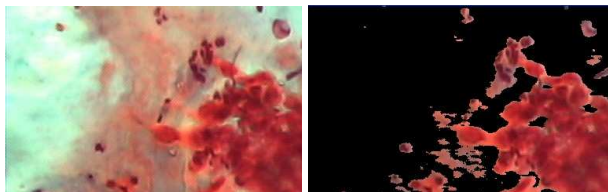


Figure 8. Segmentation in cytological image (from pap smears) using chromaticity distance measurement.

7. REFERENCES

[1] R.D. Dony and S. Wesolkowski, "Edge Detection on Color Images using RGB Vector Angles", Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering, pp. 687-692, 1999.

[2] S. Wesolkowski, R.D. Dony and M.E. jernigan, "Global Color Image Segmentation Strategies: Euclidean Distance vs. vector Angle", Neural Networks for Signal Processing IX, Proceedings of the 1999 IEEE Signal Processing Society Workshop, pp. 419-428, 1999.

[3] R Lukac et al., "Vector Filtering for Color Imaging", IEEE Signal Processing Magazine, pp. 74-86, Jan. 2005.

[4] Gonzales, R.C., Woods, R.E., *Digital Image Processing*, Addison-Wesley Publishing Company, 1992.

[5] Castleman, K.R. *Digital Image Processing*, Prentice Hall,1996.

[6] Dargham, J.A.; Chekima, A.; Lips Detection in the Normalised RGB Colour Scheme. Information and Communication Technologies, 2006. ICTTA '06. 2nd, vol. 1, 2006, pp.1546-1551.

[7] Mao, K.Z.; Peng Zhao; Puay-Hoon Tan; Supervised learning-based cell image segmentation for P53 immunohistochemistry. Biomedical Engineering, IEEE Transactions on, vol.53, no.6, June 2006, pp.1153-1163.