Técnicas de rastreamento e aplicações em análise cinemática de movimentos humanos

Pascual Jovino Figueroa Rivero

Tese de Doutorado

Instituto de Computação
Universidade Estadual de Campinas

Técnicas de rastreamento e aplicações em análise cinemática de movimentos humanos

Pascual Jovino Figueroa Rivero[1]

Fevereiro de 2004

Banca Examinadora:

- Neucimar Jerônimo Leite, Dr (Orientador)
- Roberto de Alencar Lotufo, PhD
- Luis Marcos Garcia Gonçalves,PhD
- Siome Klein Goldenstein, PhD
- Ricardo Anido, PhD
- Alexandre Xavier Falcão, Dr (Suplente)
- Sergio Augusto Cunha,Dr (Suplente)

[1]Supported in part by CNPq, under grants 141100/1998-2.

Substitua pela ficha catalográfica

Técnicas de rastreamento e aplicações em análise cinemática de movimentos humanos

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Pascual Jovino Figueroa Rivero e aprovada pela Banca Examinadora.

Campinas, 27 de fevereiro de 2004.

Neucimar Jerônimo Leite, Dr (Orientador)

Ricardo M. L. Barros, Dr (Co-orientador)

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Substitua pela folha com a assinatura da banca

Prefácio

O rastreamento de objetos a partir de imagens de vídeo é um problema que tem despertado muito interesse nos últimos anos, principalmente pela ampla gama de aplicações, tais como realidade virtual, sistemas de vigilância, robótica, análise de movimentos humanos, etc. Na análise do movimento humano as variáveis cinemáticas (por exemplo, a distância percorrida, velocidade, aceleração, etc) são parâmetros importantes para a avaliação clínica de pessoas com deficiências motoras, assim como para medir o rendimento do desempenho de atletas. Dentre as muitas formas de descrição de movimentos humanos, a análise cinemática por videogrametria tem um grande potencial de aplicação dada à simplicidade dos equipamentos (câmeras de vídeo) necessários à sua implementação e à variedade de problemas possíveis de serem considerados. Porém, a automação na determinação destes parâmetros é um problema computacional complexo devido ao grande número de fatores que dificultam a identificação e extração dos objetos de interesse.

O objetivo deste trabalho é desenvolver algoritmos de rastreamento que permitam a obtenção automática das coordenadas 2D ou 3D dos objetos de interesse, com a finalidade de quantificar os parâmetros cinemáticos do movimento. Neste trabalho, os problemas relacionados ao rastreamento de objetos são abordados a partir de duas áreas de aplicação: análise clínica (marcha, coluna, movimento respiratório), e a ánalise de desempenho dos atletas (jogadores de futebol). No primeiro caso, apresentamos algoritmos para o rastreamento dos marcadores que são fixados no corpo do sujeito nos pontos de interesse (por exemplo, as articulações). Esta abordagem leva em consideração aspectos tais como o posicionamento e quantidade de marcadores. Já o segundo problema aborda especificamente o rastreamento de jogadores de futebol durante uma partida, levando-se em conta as possíveis mudanças de iluminação, oclusão de jogadores e o uso de múltiplas câmeras com diferentes enquadramentos.

Abstract

The problem of tracking objects from a sequence of video images has been of great interest in the recent years. This interest is motivated, especially, by the wide range of applications such as virtual reality surveillance systems, robotic, human motion analysis, etc. In the human motion analysis, the kinematic variables, (e.g., the covered distance, velocities and accelerations) are important parameters for clinical evaluation of the subjects with motion disabilities, as well as measurement of motion performance of athletes. Among the different ways of human motion description, the kinematic analysis by videogrammetry represents a wide of aplication due the simplicity of the used devices (video cameras) for their implementation and the variety of problem that can be considered. However, the automatic determination of these parameters is a complex computational problem because of the great number of aspects that make difficult the identification and extraction of the objects of interest.

The aim of this work is to develop algorithms for tracking of objects allowing an automatic determination of their 2D or 3D coordinates, as well as the quantification of the kinematic variables of human motion. In this work, the problems related to tracking are associated with two applications: the clinical analysis of human motion and athlete performance analysis. In the first case, we present algorithms for tracking of markers fixed to the human body in the regions of interest (e.g., in the articulations). This approach takes into account aspect such as placement and quantity of markers. In the second case, we consider the problem of tracking soccer players during a match, taking into consideration the possible changes of illumination, players' occlusion, and the use of multiple cameras covering the whole playing field.

Agradecimentos

Acima de todo agradeço a Deus e a todas as pessoas que me ajudaram a superar as dificuldades encontradas no caminho e chegar à conclusão deste trabalho.

Ao meu orientador, Prof. Neucimar J. Leite, por seu apoio constante em todas as circunstâncias, pela paciência na correção do texto e disposição durante o desenvolvimento deste projeto.

A meu co-orientador Prof. Ricardo M. L. Barros, por sua contribuição muito importante para o sucesso deste trabalho, pelo apoio constante e por fazer possível o desenvolvimento do projeto no laboratório de Instrumentação para a Biomecânica (LIB).

A minha osita e companheira de sempre Maria del Pilar, por seu amor, carinho, compreensão e apoio, que têm sido fundamental para a conclusão deste trabalho.

A meu irmão Nilo, por ter estado sempre a meu lado apesar da distância, por sua preocupação e apoio em todo momento. Sem ele nunca teria chegado aqui.

A meu pai Jose Domingo e meus irmãos Tito, Julian e Lourdes que apesar de estarem longe sempre tem me acompanhado e apoiado em todos os momentos e têm sido minha fonte de energia.

Aos professores e amigos do LIB, René, Barreto, Sergio, e principalmente ao Prof. Euclydes que nos deixou muitos ensinamentos.

A meus amigos Milton, Alethea, Luciane, Luciana e Pedro pelo apoio, amizade e muitos anos de convívio.

Aos amigos do LIB, Tiago, Karine, Rafael, Clodoaldo, Mario, Andreia, Angelica, Amanda, Olival, Cintia, Daniela, Fernanda, Amanda, Carla e Antonio pela amizade e por fazerem do laboratório um ambiente agradável de trabalho.

A Unicamp e ao Instituto de Computação pela oportunidade de fazer os meus estudos de pos-graduação e aos profesores que contribuiram para minha formação professional.

Às pessoas do Instituto de Computação, em especial a Vera, Daniel e Roseli, e aos colegas da pos-graduação.

Aos amigos de dia a dia, Hilda, Karin, Percy, Gliseida, Rodolfo, Ernesto, Florentino, Rita, Ivete, Abdon, Honorato, Richard, Jose Maria e aos amigos do futebol da moradia.

Ao CNPq e FAEP pelo apoio financeiro. A FAPESP pelo apoio na aquisição de equipamentos.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

O crescente interesse ocorrido nos últimos anos por aplicações de visão computacional [6] tem possibilitado avanços consideráveis nesta área. Este interesse deve-se principalmente às possibilidades de sua aplicação em diferentes áreas, tais como a robótica, realidade virtual, sistemas de vigilâncias, análise de movimentos humanos, esporte, etc. A área de visão computacional aborda o problema de extração ou obtenção de informações a partir de imagens. Estas informações podem auxiliar, por exemplo, no movimento autônomo de robôs, criação de ambientes virtuais, detecção de intrusos em ambientes restritos, avaliação de pessoas com deficiências motoras, melhoria no desempenho de atletas, etc.

Com o aumento da qualidade das imagens e computadores mais potentes é possível abordar problemas cada vez mais complexos relacionados a visão computacional. Para tanto, as ferramentas de processamento de imagens tém sido, naturalmente de grande auxilio. O acesso a câmeras de vídeo de baixo custo e a imagens de boa qualidade tem motivado significantemente o desenvolvimento de aplicações nesta área.

Numa grande parte destas aplicações, o problema principal consiste em detectar e rastrear objetos de interesse. Por exemplo, um sistema de segurança deve ser capaz de identificar uma pessoa através do reconhecimento da face, forma de caminhar ou algum outro aspecto característico. Para tanto o sistema deve fazer o rastreamento da face, rastreamento dos pés da pessoa ou o rastreamento de outras partes do corpo. O rastreamento de objetos a partir de imagens de vídeo consiste em detectar o objeto de interesse e acompanhar sua localização, quadro a quadro, ao longo da seqüência.

Uma das áreas em que a visão computacional tem-se constituído numa ferramenta muito útil é área de análise de movimentos humanos, principalmente na análise cinemática cujo objetivo é quantificar as alterações na movimentação do corpo humano em diferentes atividades. Uma destas aplicações concerne, por exemplo, na análise clínica de pessoas com deficiências motoras, onde é necessário avaliar e quantificar os tratamentos aos quais estas são submetidas. Esta forma de avaliação é mais objetiva que uma simples

inspeção visual e parecer subjetivo. Porém, para que os dados possam ser realmente úteis é necessário garantir uma boa precisão e confiabilidade destes dados. Para analisar o movimento destes pacientes, geralmente são usados sistemas opto-eletrônicos que utilizam marcadores reflexivos colados ao sujeitos em pontos de articulação. Em seguida estes pacientes são filmados com câmeras de vídeo especializadas e acompanhados por um iluminador infravermelho. A refletividade dos marcadores facilita sua detecção e rastreamento durante a marcha. As coordenadas dos marcadores na imagem são transformadas em coordenadas reais usando uma prévia calibração das câmeras. Estes dados são usados, por exemplo, para calcular os ângulos entre as articulações que servem como parâmetros de avaliação da marcha. Como se pode perceber, o rastreamento correto dos marcadores, neste caso, é o fator principal para uma boa avaliação do movimento.

Uma outra aplicação de imagens de vídeo em visão computacional refere-se à análise cinemática de atletas em diversos esportes, com o objetivo de avaliar e melhorar o desempenho dos mesmos. Neste contexto, tenta-se estudar, por exemplo, o movimento de atletas com melhor desempenho para entender, do ponto de vista biomecânico, os fatores que levam a esta. Da mesma forma pode-se analisar, por exemplo, o desempenho dos jogadores de futebol em termos da distância total percorrida durante o jogo associado a variáveis fisiológicas.

Em geral, o rastreamento de objetos a partir de imagens de vídeo ou seqüência de imagens é um problema complexo, especialmente quando existem muitos objetos juntos e com características similares. Apesar da melhoria da qualidade das imagens, dependendo do ambiente de filmagem, as imagens podem sofrer alterações tanto na intensidade como no contraste, dificultando desta forma, a identificação correta dos objetos quadro a quadro, principalmente quando são empregadas câmeras de vídeo padrão nas filmagens. Um dos principais fatores que afeta a qualidade das imagens da seqüência a ser processada é a mudança de iluminação que é particularmente significativa em ambientes externos não controlados.

O problema de rastreamento de objetos a partir de imagens de vídeo é tratado, geralmente, em duas etapas. A primeira consiste na segmentação dos objetos de interesse e, a segunda, no rastreamento propriamente dito. O objetivo da segmentação é extrair ou separar as regiões da imagem que representam os objetos de interesse. Desta forma, durante o rastreamento, são usadas apenas as informações extraídas na etapa de segmentação que, conseqüentemente é muito importante para garantir o sucesso do rastreamento.

Neste trabalho, abordarmos alguns destes problemas a partir de duas aplicações: a primeira refere-se ao rastreamento de marcadores para a análise de movimentos humanos e, a segunda ao rastreamento de jogadores de futebol durante uma partida. Em ambos os casos, o objetivo principal é determinar a posição dos objetos de interesse (marcadores e jogadores) no espaço 2D ou 3D, visando a análise cinemática do movimento.

O enfoque principal deste trabalho concentra-se desenvolvimento de sistemas ou ambientes que permitam, de forma geral e simples, o emprego de ferramentas de processamento de imagens e de visão computacional para a determinação automática da posição de objetos em movimento a partir de uma seqüência de imagens. Cada Capítulo desta tese é apresentado em forma de artigo, nos quais são abordados problema específicos à área de aplicação.

No Capítulo 2 deste trabalho é abordado o problema relativo ao rastreamento isolado ou em grupo de objetos com características similares, como é o caso de marcadores empregados na análise clínica de marcha. Estes marcadores são objetos colados ou fixados em pontos de articulação de pessoas e que auxiliam na determinação de sua posição real no espaço 3D e, conseqüentemente, na análise quantitativa da marcha. Embora o rastreamento de objetos isolados seja um problema relativamente, mais simples, existem aspectos tais como interferências, posicionamento dos objetos muito próximos ou até mesmo de oclusões que podem dificultar este rastreamento. Neste sentido, é importante, utilizar todas as informações disponíveis sobre o modelo do rastreamento e ponderar-las de acordo com sua importância.

O algoritmo proposto aqui, baseia-se em segmentação, predição e casamento de padrões. O objetivo da segmentação é extrair os objetos de interesse, tendo como resultado um conjunto de regiões conexas etiquetadas. Esta etapa permite selecionar a seqüência de ferramentas que define um algoritmo de segmentação de acordo as particularidades da aplicação. O próximo passo consiste em usar o casamento de padrões para comparar o modelo do objeto com as regiões extraídas durante a segmentação. Finalmente é selecionada a região que dê maior índice de correlação. A predição é importante para limitar a região de busca na imagem. Em situações em que existem objetos muito próximos, a distância ao ponto predito pode ser considerada como um critério de decisão.

Os sistemas comerciais [23, 38, 66] para a análise tridimensional de movimento humanos, têm considerado essencialmente filtros passa - baixas (limiarização) na segmentação dos marcadores já que, ao se considerar luzes infravermelhas, os marcadores reflexivos aparecem com maior intensidade nas seqüências de imagens.

O rastreamento de movimentos humanos também tem sido abordado ainda através de modelos em forma de segmentos [2, 43], cilindros [60, 2], elipsóides [28, 53], etc. com objetivo de identificar o movimento de partes do corpo humano visando o reconhecimento automático de gestos, atividades e tipos de movimentos. Uma análise cinemática do movimento humano é uma operação muito mais complexa dada a necessidade de se utilizar modelos tridimensionais com muitos graus de liberdade, para se obter dados precisos e reais deste movimento.

A segmentação de objetos a partir de imagens vídeo é uma etapa fundamental nas aplicações de visão computacional e constitui a base para as etapas posterior envolvendo,

por exemplo, rastreamento, reconhecimento e análise do movimento. Esta segmentação tem sido abordada de diferentes formas, dentre elas, a mais simples e bastante utilizada, quando do emprego de câmeras fixas, baseia-se na diferença entre imagens da cena. De modo geral, o método consiste da diferença entre a imagem atual e uma imagem do fundo sem os objetos em movimento. Este tipo de abordagem não é aplicável para o rastreamento dos marcadores já que estes se deslocam juntamente com o corpo humano. Uma das dificuldades do método das diferenças é que nem sempre podemos dispor de uma imagem de fundo e, portanto, é necessário extrair esta imagem a partir da própria seqüência, eliminando os objetos em movimento. A imagem de fundo nem sempre é constante. Alguns objetos que se encontram em movimento podem se transformar em objetos de fundo quando deixam de se movimentar por um longo tempo. Também podem ocorrer mudanças na iluminação, o que implica na necessidade de uma atualização constante desta imagem de fundo. Os métodos comumente usados para esta finalidade são os métodos estatísticos de atualização [59, 27, 31, 19] que levam em consideração a média e o desvio padrão das mudanças para identificar os pixels que pertencem ao fundo e atualizar a imagem de referência. No entanto, estes métodos precisam gerar uma imagem de fundo inicial confiável, e qualquer erro na identificação dos pixels pode gerar uma imagem de referência incorreta. Mudanças bruscas de iluminação em ambientes não controlados são exemplos de problemas que podem ocasionar a definição de um falso fundo em seqüência de imagens.

No Capítulo 3 deste trabalho abordamos o problema de extração do fundo em ambientes com mudanças brusca de iluminação. O método proposto considera um conjunto de imagens da seqüência e uma função temporal para cada pixel desta seqüência. A esta função é aplicada uma seqüência de operações morfológicas com o objetivo de eliminar os objetos em movimento. Alguns parâmetros estatísticos são considerados aqui visando uma redução do tempo de processamento. O método foi aplicado em seqüências de imagens de um jogo de futebol filmado com 4 câmeras, durante um dia ensolarado e com presença de nuvens, o que provoca mudanças bruscas de iluminação em alguns intervalos de tempo.

O rastreamento de atletas com o objetivo de quantificar o seu rendimento assim como o de melhorar o seu desempenho tem sido de grande interesse tanto de pesquisadores quanto de técnicos e treinadores. Para esta finalidade, câmeras de vídeo e sistemas baseados em sensores têm sido empregados na localização da posição de atletas. O uso de imagens de vídeo tem a grande vantagem de não interferir na movimentação destes atletas, além de fornecer uma informação visual mais ampla sobre o movimento realizado. Em particular, a análise da movimentação de atletas em esportes coletivos tem despertado um grande interesse, especialmente no caso futebol, por ser um dos esportes mais populares do mundo. O interesse não esta centrado apenas em quantificar o esforço e o desempenho desde o

ponto do vista biomecânico e fisiológico, mas também pela possibilidade de estudar e analisar o posicionamento tático dos jogadores em campo.

As primeiras abordagens visando quantificar a movimentação dos jogadores de futebol foram através de estimativas [46, 72]. Posteriormente, foram consideradas técnicas baseadas em geometria projetiva para se obter informações mais precisas sobre a posição dos jogadores. No entanto, a extração automática destas informações tem-se constituído num problema complexo. Um dos primeiros trabalhos, visando a análise automática do comportamento em grupo de jogadores de futebol, foi realizado por Taki et. al [68]. Para tanto, um jogo foi filmado com várias câmeras e apenas jogadores isolados foram rastreados de forma automática. Nas demais situações, as posições de cada jogador foram localizadas manualmente. Os trabalhos posteriores relacionados com o rastreamento de jogadores [73, 45, 36, 69] têm se concentrado, principalmente, nas aplicações envolvendo animação e reconstrução de jogadas, a partir de imagens transmitidas pela televisão ou então filmadas cobrindo apenas a área do gol. Os métodos de rastreamento proposto apresentam apenas testes em trechos curto de seqüência de imagens.

No Capítulo 4 desta tese é abordado o problema do rastreamento de jogadores de futebol. A partir das informações extraídas na segmentação, o rastreamento é abordado usando a representação em forma de grafos. O grafo é construído considerando os nós como os blobs definidos numa etapa de segmentação e as arestas, a distância entre os nós de dois quadros consecutivos. Os jogadores são rastreados percorrendo-se o grafo na busca de caminhos mínimos. Além da distância entre blobs, a cor e forma destes são explorados na determinação correta da trajetória de um jogador. As oclusões são tratadas, tentando subdividir os blobs em dois ou mais novos blobs, a partir da informação armazenada no grafo. O método foi testado em jogos oficiais do campeonato brasileiro filmados com 4 câmeras e alguns resultados da aplicação são apresentados.

O método para o rastreamento de jogadores pode ser facilmente adaptado a outras aplicações tais como, rastreamento de jogadores de futebol de salão, tênis, handebol, basquetebol, etc. Em geral, apesar das particularidades das diferentes aplicações, o rastreamento de objetos têm características em comum que devem ser exploradas por um sistema ou ambiente flexível que permita o aproveitamento de módulos e a montagem de algoritmos específicos a uma dada aplicação.

No Capítulo 5, apresentamos o ambiente visual desenvolvido que permite a construção flexível e modular de diferentes algoritmos de processamento de seqüência de imagens. Os algoritmos são construídos a partir da seleção de um conjunto de ferramentas disponíveis para este fim e da configuração dos parâmetros das respectivas funções, de acordo com a aplicação específica. Neste Capítulo são descritos, ainda, os detalhes de implementação, dando importância aos mecanismos de controle de laços que permitem a construção de procedimentos com iterações aninhadas.

Finalmente, O Capítulo 6 da tese faz considerações finais sobre o trabalho e comentários sobre extensões futuras.

Este trabalho foi realizado no âmbito de uma cooperação estabelecida entre o Laboratório de Instrumentação para Biomecânica da Faculdade de Educação Física e o Instituto de Computação da UNICAMP.

Capítulo 2

Um sistema flexível para rastreamento de marcadores usado em análise de movimentos humanos

Neste Capítulo apresentamos o artigo titulado "A flexible software for tracking of markers used in human motion analysis" publicado na revista *Computer Methods and Programs in Biomedicine* [24]. O principal objetivo deste trabalho foi desenvolver algoritmos de rastreamento de marcadores que permitam a análise de diferentes tipos de movimentos humanos. Uma das motivações foi a possibilidade de desenvolver sistemas de baixo custo baseado em vídeo e com ampla gama de aplicações, ao contrário dos sistemas óptico-electrônicos comerciais que realizam o rastreamento automático de marcadores em ambientes controlados e aplicações restritas. Por exemplo, a maioria dos sistemas para análise de marcha considera apenas os membros inferiores, devido à dificuldade no rastreamento simultâneo de um grande número de marcadores.

A idéia inicial de desenvolver sistemas de baixo custo baseados em videogrametria, usando câmeras de vídeo padrão e ferramentas de processamento de imagens, foi descrita no trabalho de Barros [16]. Posteriormente, algoritmos mais eficientes foram propostos, baseados em Morfologia Matemática, para melhorar o rastreamento dos marcadores [25, 26]. No trabalho de Barros et. al. [8] é discutida uma aplicação deste sistema e a avaliação dos algoritmos, em termos de precisão, em que a marcha de dois sujeitos foi comparada, um normal e um outro com histórico de intervenção cirúrgica.

O artigo apresentado neste capítulo descreve uma parte do sistema referente ao rastreamento dos marcadores, aperfeiçoado e extendido e que leva em conta outras áreas de aplicação. O ambiente proposto considera dois tipos de configurações: o primeiro, trata o problema de rastreamento de marcadores isolados e, o segundo, o caso de marcadores muito próximos, cujo rastreamento é feito em grupo. Os algoritmos de segmentação em

ambos os casos são os mesmos, defirindo aqueles referente à etapa de reconecimento de formas. No rastreamento de marcadores isolados é considerado o casamento de padrões e no rastreamento em grupo, o casamento por emparelhamento considerando as distâncias entre marcadores de quadros consecutivos. O Filtro de Kalman é usado na predição da posição dos marcadores nos quadros sucessivos e na definição as regiões de busca.

Os algoritmos de segmentação são aplicadas sobre imagens em níveis de cinza e exploram informações sobre intensidade, contraste e forma na extração das regiões de interesse correspondente aos marcadores. Para tanto são empregados transformações de Morfologia Matemática tais como, abetura, fechamento, dinâmica, linha divisora de água (watersheds), etc. As definições destas funções são apresentadas no início do artigo. A segmentação melhora a precisão dos dados que, combinados com técnicas de casamento de padrões, aumenta a robustez e flexibilidade do sistema.

Os algoritmos descritos neste artigo podem ser considerados em aplicações envolvendo por exemplo, análise de padrões respiratórios em adultos praticantes de yoga [9], análise das curvaturas da coluna durante a marcha [11], análise de marcha e modelos de representação dos membros superiores [4, 5], análise de volumes respiratórios [12, 61], análise da flexão do joelho no salto vertical [50] e análise dos dedos da mão durante a presão palmar [21].

A generalização do programa possibilita o emprego do sistema em outros tipos de rastreamento, considerando-se os objetos de interesse como marcadores. como é o caso, por exemplo, do rastreamento de jogadores de futebol em situações em que estes se encontram isolados no campo ou em situações simples de oclusão. Este modelo foi testado ainda no rastreamento de ratos de laboratórios submersos em água e que se movimentam à procura de uma plataforma (teste de water maze).

Finalmente, como veremos no Capítulo 4, os conceitos deste sistema são estendidos na abordagem do problema de rastreamento, em situações mais complexas envolvendo oclusões múltiplas e objetos com características diversas.

A flexible software for tracking of markers used in human motion analysis

P.J. Figueroa N. J. Leite R. M. Barros

Institute of Computing
State University of Campinas
{pascual,neucimar}@ic.unicamp.br

Faculty of Physical Education
State University of Campinas
ricardo@fef.unicamp.br

Abstract

In this work, we present a software for the tracking of markers used in human motion analysis. This software is based mainly on image sequences captured by video cameras and on image processing and computer vision tools. Unlike the optoelectronic systems, which record only the coordinates of the markers, a video-based system offers more visual information and flexibility which can be exploited in different applications. However, it needs a more complex tracking procedure concerned with the extraction and identification of the used markers.

The tracking module presented here is divided into the following three procedures: segmentation, matching and prediction. The segmentation consists in extracting the objects of interest (markers). The matching is used to find the correspondence between the extracted objects in two consecutive frames. The prediction is important to limit the region of processing, thus reducing the execution time. Some results of the automatic tracking is presented together with their application in human motions analysis.

2.1 Introduction

Analysis of human motion is of great importance in different areas of medicine and human science. In clinical gait analysis, for example, the temporal information about the position and orientation of the patient's joints may be useful in determining abnormalities [74]. In orthopedics, the range of motion information helps in the evaluation of the prosthetic join replacement and physical therapy of joint disease [58]. In neurology, the early diagnosis of cerebral lesions appears to be possible by detecting a slight deviation from the normal patterns which can not be evident by a simple visual inspection [22]. Postural analysis during motion can be studied from the 3-D representation of the column through the position of the markers fixed to the spinal process [15]. Athletic performance can also be improved by breaking down the movement into elementary components and by identifying

suitable reference models arising from the observation of the outstanding athletes [55]. For these reasons, there is a great interest in developing measurement techniques that allow a more accurate and automatic analysis of the human movement. In particular, commercial optoelectronic systems, based on markers and TV cameras, have been extensively used to record 3-D kinematics of the human motion [58]. Markers can be defined as special objects that are attached or fixed to the human body, helping to track the movement of its interest points.

These systems can be divided into active and passive systems, depending whether they use active or passive markers. The approach of such systems for tracking considers dedicated hardware implementing basic computer vision techniques. Systems based on active markers, such as SELSPOT [66], OPTOTRACK [42], COSTEL [44], etc., use a certain number of light emitting diodes (LED) attached to the body. Each LED is connected with wiring to a led control unit transported by the subject and activated stroboscopically by a cable or telemetric multiplexed signal. The extraction of the markers in active systems is almost straight since the markers and the background present high contrast. The tracking is simple because only one LED is activated at a time. However, the applicability of these systems is reduced by the mechanical restraints imposed by the LED-wiring attached to the subject, the necessity of a controlled environment and the limitation of the number of markers due to the stroboscopic timing. Some of these limitations may be avoided by the use of passive markers.

Passive infrared systems such as ELITE [23], MaxReflex [38], and VICON [66] operate through the recording of reflected light, using wireless retro-reflective markers in combination with infrared illumination. In such a case, problems related to wires linked to the subject and managed by a central control unit are avoided. Although the images obtained by infrared cameras have a high contrast between the markers and the background, the presence of many markers simultaneously in the images requires the use of labeling algorithms yielding a unique identification of the markers along the image sequences. This process can fail in some situations, for example, when the markers are placed in anatomical points close to each other. The main advantage of these commercial optoelectronic systems is their real-time processing capability. Their main drawback lies in the limitations concerning the experimental setup. For example, in order to guarantee the automatic tracking, some restrictions are imposed to number, proximity and size of the markers, as well as to data acquisition environment, camera view field, body segments and the movements supposed to be analyzed. In gait analysis, for example, most of the systems consider only the lower part of the body, with only one degree of freedom to describe the foot orientation. Another important aspect to take into account is that the infrared-based systems record only the coordinates of the markers, thus loosing the potential pictorial information that can be useful in a qualitative analysis of the movement.

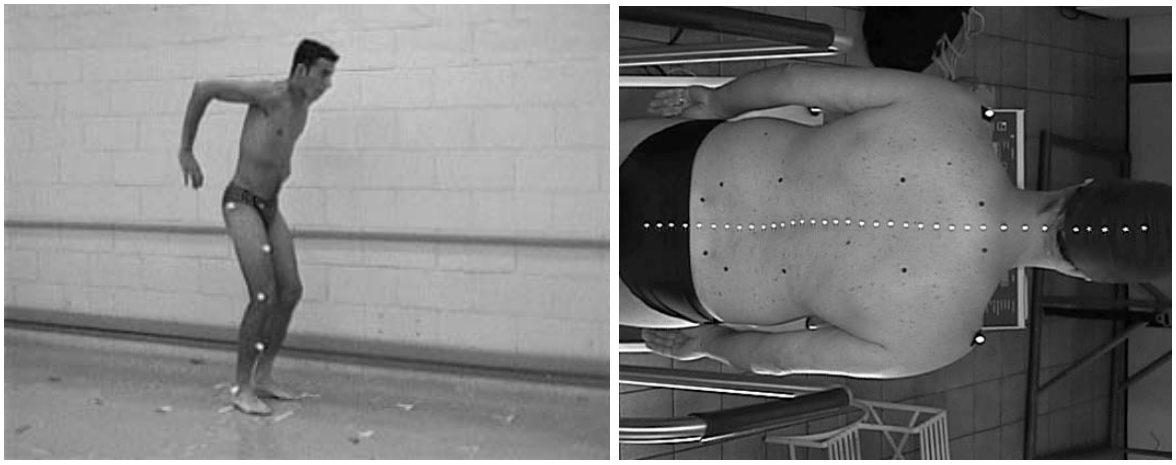
With the development of high-speed personal computers capable of processing and storing a great amount of information, it is already possible to develop low-cost systems based on conventional CCD video cameras. The system described in [74] considers conventional video camera and uses the Kalman Filter [32] and a template matching algorithm [64] for tracking passive markers. However, the environment considered by such a system must be controlled for the marker detection and identification, (e.g., the patient must wear a skin tight dress of well contrasted color). A full automatic tracking of markers is a challenging problem, even when we consider those systems based on infrared data where the contrast between the image background and the markers can help in their automatic recognition. This problem becomes more important when we take into account, for example, the variability of environment conditions and the different levels of complexity of the studied movements. The purpose of the present work is to describe a software for tracking markers considering, mainly, morphological region-based segmentation, template or distance-based matching and Kalman filter prediction. The main advantage of such a software is its flexibility to define an application-dependent framework for tracking markers, using basic image processing tools. This framework allows different definitions of algorithms, depending on the markers characteristics and the application features.

This work is organized as follows: the software description is introduced in the next Section, taking into account each step of the overall processing. Some applications and their results are illustrated in Section 3. The hardware and software specifications of the system are given in Section 4. Finally, some conclusions are drawn in Section 5.

2.2 Software description

The input of this tracking system is a sequence of images (video in AVI format or sequence of bitmaps) captured by a video camera. The output are the coordinates of the markers position at each frame. Examples of images of two different input sequences are shown in Fig. 2.1. Depending on the analyzed movement, some markers may be more suitable than others, if we take into account, for example, precision, handling (easy to mark and remove), and mainly efficiency (easy to track). Naturally, a flexible system must be able to deal with most of these properties. Fig. 2.1(a) shows an example of spherical passive non retro-reflexive markers for the analysis of jumping. Analogously, Fig. 2.1(b) shows the use of planar retro-reflective passive markers placed very close to each other.

Although the use of high-speed personal computers makes possible the definition of complex image processing tools, it is necessary to limit the image regions to be analyzed in order to get near real-time processing. Thus, we consider a region-based tracking method with two types of configurations. The first one concerns the tracking of isolated markers which are tracked independently. The second type considers the case in which the markers



(a) (b)

Fig. 2.1: Examples of images of two different input sequences with different types of markers. a) spherical passive markers.

are very close to each other. In such a case, these markers are grouped together into a set (defining a segmentation region) and are finally tracked.

In the tracking of isolated markers (Fig. 2.2), the markers model is defined according to the first frame or updated during the sequence processing. Starting from the second frame, a search region is selected for segmentation. The result of this processing is considered during the matching of the marker model on the selected region image. The output of the matching are the coordinates of the marker position. This result is used to update the prediction filter and to predict the position of the markers at each frame. Further, this information is used to define the search region at the next frame. This procedure is repeated for each marker of the sequence.

In the tracking by sets of markers (Fig. 2.3), the markers of a region, specified by the user (Section 2.2.4) in the first frame, are extracted, based on the segmentation method described later (Section 2.2.1), and their position is calculated. The search region for the next frame is defined here by using this position information. Later, the segmentation is processed (Section 2.2.1) and the output image is considered in a labeling algorithm in which the segmented regions, corresponding to the different markers, are labeled, followed by the computation of the center of these regions. The coordinates of the markers position are then considered in the matching step to determine the correspondence between markers at two consecutive frames. The matched coordinates are used to update the prediction filter and to define the center of the search region at the next frame.

The tracking initialization module shown in Fig. 2.2 and Fig. 2.3 will be discussed in Section 2.2.4. In the following, we introduce the segmentation model considered in this work.

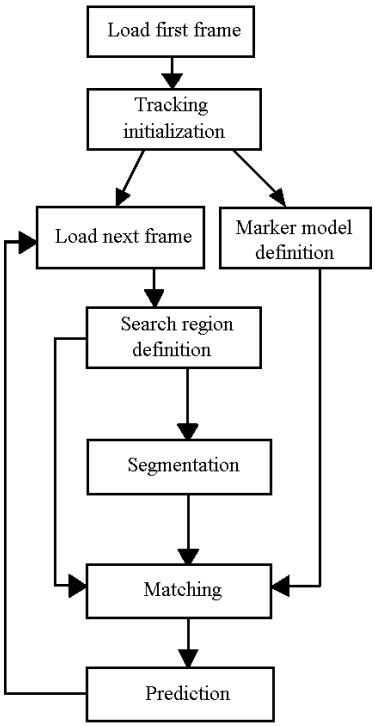


Fig. 2.2: Sequence of operations for tracking isolated markers.

2.2.1 The basic segmentation operations

The segmentation step attempts to extract regions of the image corresponding to the markers in the original scene. Here, this segmentation is based on image processing tools, mainly on mathematical morphology transformations which are represented by the following basic operations [63].

Dilation and erosion: these two morphological operations constitute the basis of more complex morphological transformations and can be defined as follows [30]:

Let f be a function representing the image as a topographic surface, and g a planar structuring element (a small image whose height is zero), so that: $f : F \rightarrow Z$, $g : G \rightarrow Z$ and $F, G \subset Z^2$.

The dilation of f by g is given by:

$$(f \oplus g)(x) = \max_{z \in G,\, x-z \in F} \{f(x-z)\}. \tag{2.1}$$

and the erosion of f by g is:

$$(f \ominus b)(x) = \min_{z \in G,\, x+z \in F} \{f(x+z)\}. \tag{2.2}$$

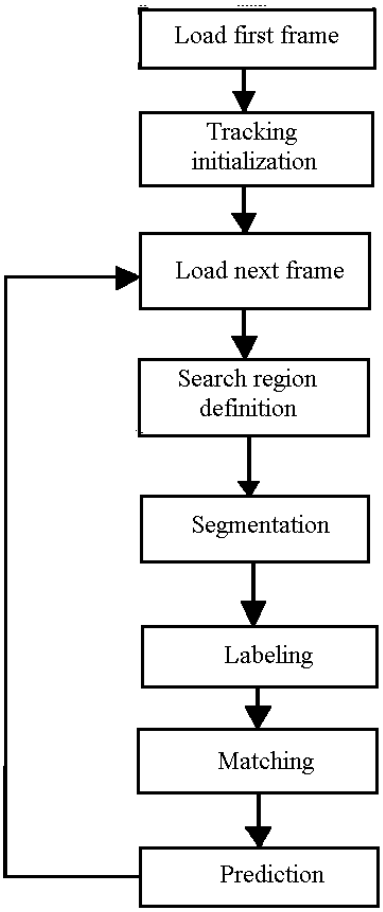


Fig. 2.3: Sequence of operations for tracking sets of markers.

Opening and closing: by interactively applying dilation and erosion, one can eliminate details of an image, which are smaller than the structuring element, without affecting its global geometric features. For example, opening smoothes contours by breaking narrow isthmuses and eliminating small islands of an image. This operation is given by:

$$f \circ g = (f \ominus g) \oplus g \tag{2.3}$$

On the other hand, closing smoothes contours by filling narrow gulfs and eliminating small holes. This operation is defined as:

$$f \bullet b = (f \oplus g) \ominus g \tag{2.4}$$

Watershed: an intuitive definition of the morphological watershed transformation [10] may be given by considering, as before, an image f as a topographic surface. This

surface presents regional minima, as shown in Fig. 2.4. Suppose these regional minima are pierced and the surface is submerged progressively into water, so that the level is the same at the entire surface. During this process of flooding, dikes are built in order to avoid the join of water coming from two different minima. At the end of flooding, the lines formed by the set of dikes constitute the watersheds of image f.

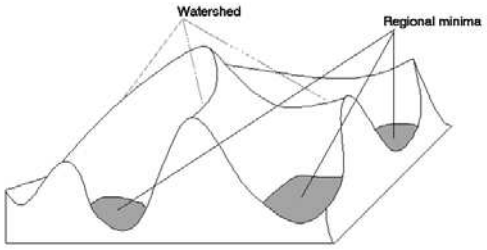


Fig. 2.4: An intuitive definition of watershed.

Dynamic of regional maxima: dynamic [29] is a measure of contrast of the image structures. For a regional maximum M (Fig. 2.5), this measure is given by

$$dyn(M) = f(M) - \max\{\min\{f(x), x \in P\}\}, \tag{2.5}$$

where
$P = \{(p_0, p_1, ..., p_n)\},\ p_0 \in M \wedge f(p_n) > f(p_0)$

Informally, the dynamic is defined as the minimal uneven obtained in the path starting from the point M until another regional maximum higher than M is reached. The dynamic of regional minima can be easily defined in an analogous way by considering the inverse of the original image. The inversion operation, k , for a 256 gray-scale image can be defined as:

$$k(x) = 256 - f(x) \tag{2.6}$$

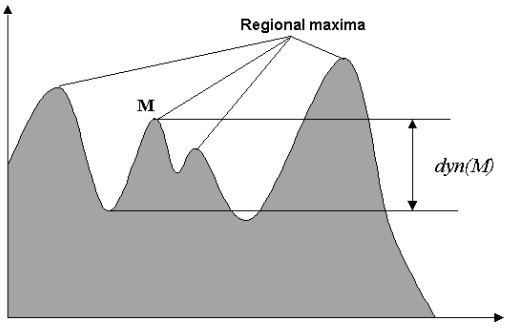


Fig. 2.5: Definition of dynamic for the regional maximum M.

The segmentation method

Fig. 2.6 illustrates an algorithm for segmentation based on the above operations. This algorithm includes the following steps:

Step 1: Pre-processing (Fig. 2.7b). This step consists in processing the selected search region of the image using morphological operators such as dilation, erosion, opening etc. The aim here is to enhance the peaks of the image representing the markers. Furthermore, one can fill the small 'holes' in the markers generated, for example, by the analog-digital conversion, insufficient spatial resolution of the cameras, and shadows. The original shape of the markers can be partially recovered using the dilation or erosion operator. Fig. 2.7b considers the inversion transformation of the original image for the implementation of the dynamic of its regional minima.

Step 2: Gradient detection (Fig. 2.7c). The contours of the image obtained at step 1 are detected using the morphological gradient given by :

$$grad = (f \oplus g) - (f \ominus g) \tag{2.7}$$

Step 3: Dynamic computation (Fig. 2.7d). The dynamic of the regional minima of the image from step 1 is computed, and some regions having the highest dynamic values are selected. The number of these selected regions is a parameter given at the initialization step and must be, at least, equal to the number of tracked markers.

Step 4: The watershed segmentation (Fig. 2.7e). The watershed of the gradient image obtained at step 2 is computed based on the set of regions obtained at step 3 (the regions pierced in the flooding process are those defined by this set). Step 5: Contours selection (Fig. 2.7f). Finally, some regions are filtered according to size and/or shape parameters (e.g., perimeter, surface, elongatedness, etc) related to the morphological structures of the markers.

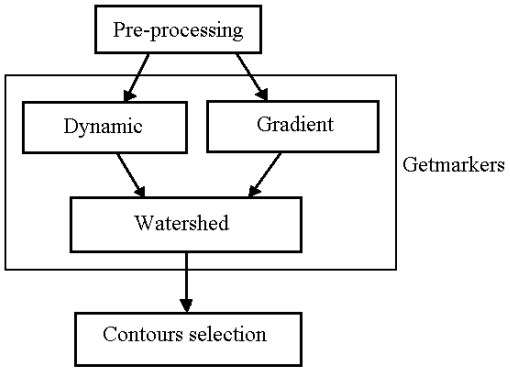


Fig. 2.6: An example of a segmentation algorithm.

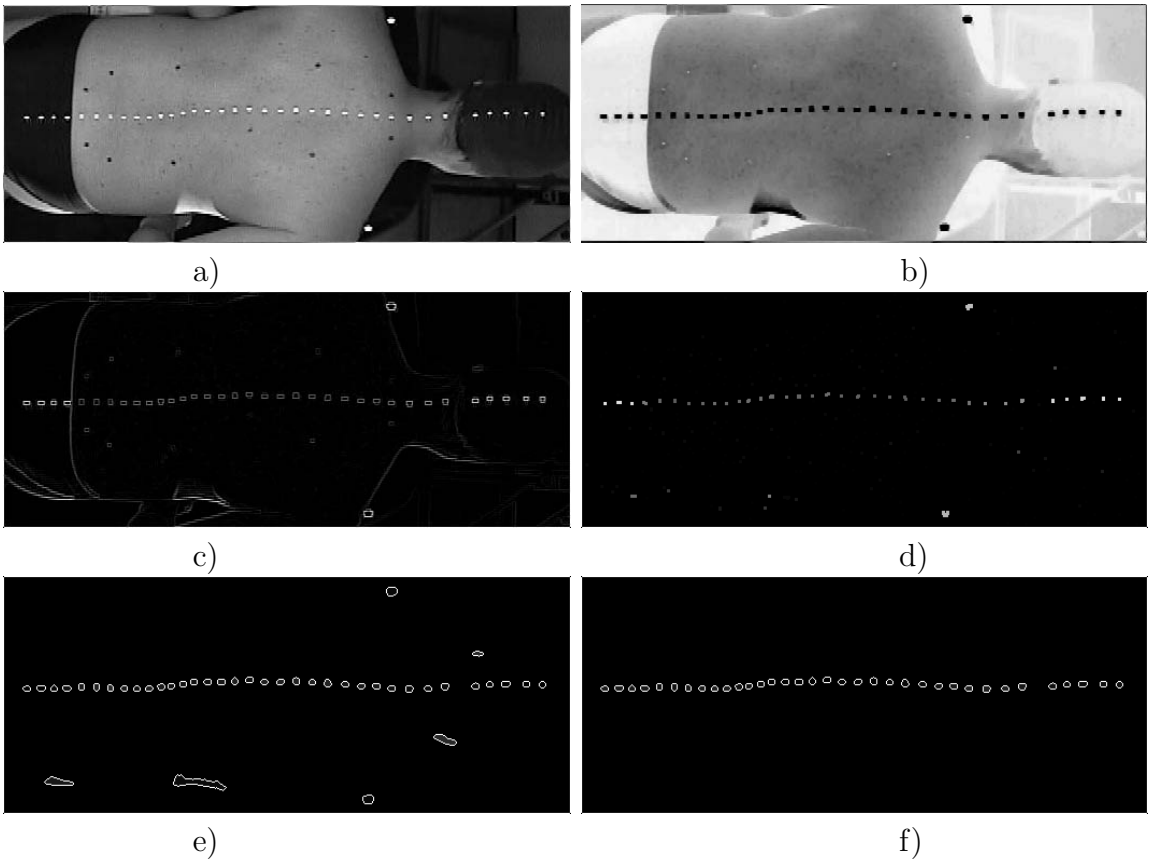


Fig. 2.7: Example of image segmentation. a) selected region from original image. b) pre-processing (inversion followed by an erosion). c) gradient detection. d) dynamic computation. e) watershed segmentation. f) contour selection.

2.2.2 Matching

Matching consists in finding the correspondence of markers between two frames. In this system, two kinds of matching are available. One concerned with the isolated marker and based on the intensity (gray-scale values) of the image pixels, and the other one taking into account the distance between markers and concerned with the matching of sets of markers.

The matching of isolated markers based on the gray-scale values of the pixels is given by the cross-correlation function between the marker model and a portion of the image corresponding to the search region. This is also called template matching [64], in the pattern recognition context, and is defined, for each point (i, j) of the image, as:

$$C_{i,j} = \frac{\sum_{m=-M}^{M}\sum_{n=-N}^{N}(f_{i+m,j+n} - \overline{f}_{i,j})(h_{k,l} - \overline{h})}{\sqrt{\sum_{m=-M}^{M}\sum_{n=-N}^{N}(f_{i+m,j+n} - \overline{f}_{i,j})^2}\sqrt{\sum_{m=-M}^{M}\sum_{n=-N}^{N}(h_{m,n} - \overline{h})^2}} \tag{2.8}$$

where f is the search region image and h is the model of the marker of size $(2M + 1)$x$(2N + 1)$ pixels. The values $\bar{f}$ and $\bar{h}$ are the mean of f and h, respectively.

The matching function is applied to the significant regions extracted from the segmentation algorithm. This local aspect reduces the processing time of the matching step. Finally, regions of high correlations are selected corresponding to the expected position of the markers.

In case of tracking sets of markers, it is difficult to find the right correspondence using template matching only. Thus, in such a case, the tracking here consists in solving the assignment problem in which the matching can be computed by finding the minimal distance between the positions of the markers in two consecutive frames. This can be stated as follows.

Let d_{rs} be the distance between marker r found at frame t and marker s found at frame $t+1$, along the image sequence. In such a case, we need to find the values $x_{rs} \in \{\}$ that minimize the equation

$$\sum_{r=1}^{R}\sum_{s=1}^{S} d_{rs}x_{rs} \tag{2.9}$$

with the following conditions:

$$\sum_{r=1}^{R} x_{rs} = 1, \quad s = 1, ..., S \tag{2.10}$$

and

$$\sum_{s=1}^{S} x_{rs} = 1, \quad r = 1, ..., R \tag{2.11}$$

Where R and S are the number of markers detected at frame t and frame $t+1$, respectively.

The solution of the above problem can be found in polynomial time ($O(n)$)using the Hungarian algorithm [54], for example. Also, the processing time can be reduced by eliminating some improbable matching characterized by large distance between two markers.

2.2.3 Prediction

In order to reduce the search region, it is necessary to define mechanisms to predict the possible position of the markers at the successive frames. For this purpose, we can use a simple extrapolation or a more complex function based on the Kalman filter [74, 32].

The Kalman filter is a recursive predictive update technique used to determine the correct parameters of a process model. Given some initial estimates, it allows the parameters of a model to be predicted and adjusted with each new measurement, providing an estimate of errors at each update. Its ability to incorporate the effects of noise (measurement and modeling) has made the Kalman Filter very popular in computer vision tracking applications. For the tracking of the markers this filter can be modeled as:

$$x(t+1) = \Phi(t)x(t) + v(t) \tag{2.12}$$

$$z(t+1) = H(t)x(t) + w(t) \tag{2.13}$$

where $x(t)$ represents the vector containing the parameters at time t . These parameters are the coordinates x,y of the marker position and their respective velocities $\dot{x}$,$\dot{y}$ i.e., $x(t) = [x, \dot{x}, y, \dot{y}]$. The measurement vector, $z(t)$, contains the coordinates of the marker position, $z(t) = [x, y]$. $\Phi(t)$ represents the state transition matrix from $x(t)$ to $x(t+1)$ and $H(t)$ is the measurement matrix representing a noiseless connection between $x(t)$ and $x(t+1)$. For our tracking purpose, they can be defined as:

$$\Phi(t) = \begin{bmatrix} 1 & \delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

where δt represents the time interval between two frames. $v(t)$ and $w(t)$ are modeled as zero mean white noise with correlation array $Q(t)$ and $R(t)$, respectively.

The following equations, derived as a solution from the process and measurement equations (Eqs. 2.12 and 2.13), are used to compute the predicted state, $\hat{x}(t+1|t)$, for $x(t+1)$ and the error covariance $P(t+1|t)$

$$\hat{x}(t+1|t) = \Phi(t)\hat{x}(t) \tag{2.14}$$

$$P(t+1|t) = \Phi(t)P(t)\Phi^T(t) + Q(t) \tag{2.15}$$

The values $\hat{x}(t)$ and $P(t)$ are updated using the following equations:

$$\hat{x}(t) = \hat{x}(t|t-1) + K(t)[z(t) - H(t)\Phi(t-1)\hat{x}(t-1)] \tag{2.16}$$

$$P(t) = P(t|t-1) - K(t)H(t)P(t|t-1), \tag{2.17}$$

where $K(t)$ is the Kalman gain:

$$K(t) = \frac{P(t|t-1)H^T(t)}{H(t)P(t|t-1)H^T(t) + R(t)} \tag{2.18}$$

Before showing some tracking results, we discuss some basic aspects of the system interface.

2.2.4 Interface description

Fig. 2.8 shows the interface of the proposed software. It is configured to load up to six sequences of images (six cameras) and to perform the tracking independently. The toolbars in the interface window are used to track the markers, correct the measurements and define the markers model.

Before starting the tracking process, the necessary parameters for each step are initialized through the dialog box illustrated in Fig. 2.9. It is also defined whether the tracking is by individual markers (Fig. 2.2) or by set of markers (Fig. 2.3). If the tracking is by set of markers, an initial region encompassing the markers of interest is defined by a rectangle on the image. If the tracking is by individual markers, the model of these markers is given using some available tools. The initial searching area is also defined as well as the number of markers to be tracked.

Fig. 2.10 shows a dialog for selecting the sequence of operations for segmentation. In general, from the set of available operators, one can select any sequence of transformation and its corresponding parameters to build the specific segmentation model. This dialog interface also allows the test of each operation before starting the tracking process. The Getmarkers operation indicated in Fig. 2.10 corresponds to the set of Dynamic, Gradient, and Watershed operations shown in Fig. 2.6.

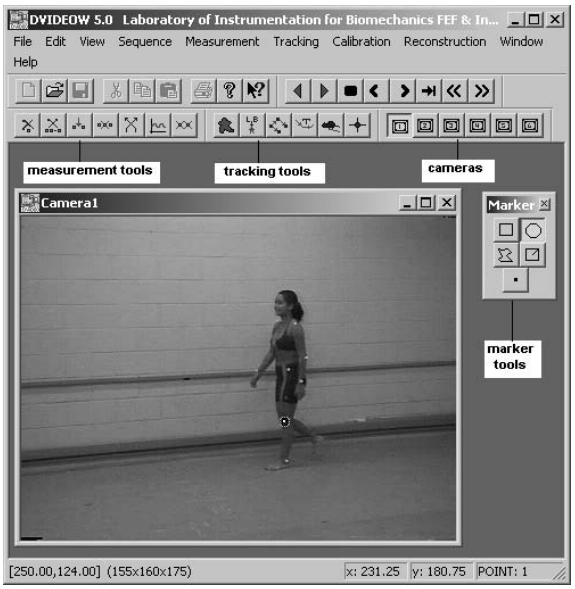


Fig. 2.8: The visual interface of the proposed system.

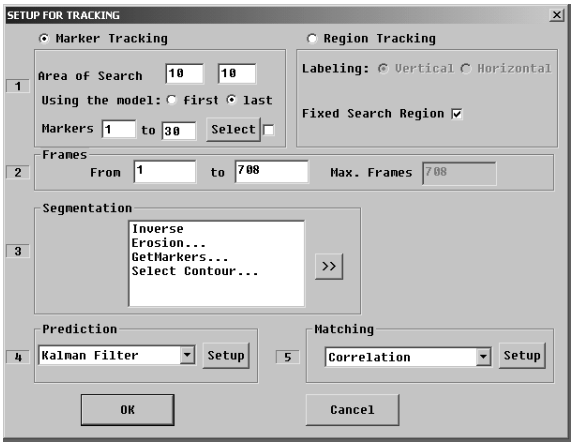


Fig. 2.9: Dialog for selecting and setting up the parameters for tracking.

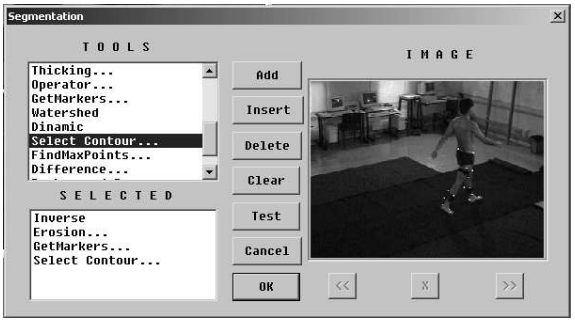


Fig. 2.10: Dialog for building the segmentation algorithm.

2.3 Results and Discussion

The software includes the implementation of the camera calibration based on a direct linear transformation method [1]. The calibration parameters are used for the 3D reconstruction of the markers. This tracking environment was successfully tested in many applications, such as jumping, breathing, running, walking etc. Two of these applications are illustrated here.

The first example (Fig. 2.11) shows the tracking of circular reflexive markers (adhesives of 5mm in diameter) for the analysis of the spine during a walking on a treadmill. In this work, two digital cameras filmed the experiment and 31 markers were attached to the skin along the spine. The size of the image captured by the digital cameras was 720x480 pixel in NTSC format and the frame rate was 60 frames/s. This feature was obtained by deinterlacing [18] the frames through the software tools.

The use of reflexive markers needs a lighting directed to them. The markers were tracked successfully using the algorithm for tracking by set of markers. As a result, it is shown the stick figures of the spine, during 50 frames, obtained from the reconstructed 3D coordinates of the markers (Fig. 2.12).

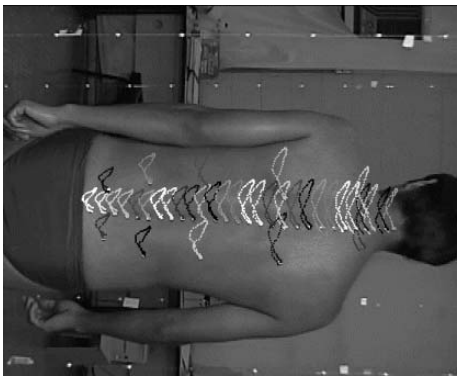


Fig. 2.11: Tracking of planar reflexive markers for the analysis of the spine during movement.

The second application is related to the tracking of spherical reflexive markers (15mm in diameter) used for the gait analysis (Fig. 2.13). This is a more complex situation due to the range of the motion and the proximity of the markers which can be occluded. For a 3D reconstruction, the movement was filmed using six digital cameras in order to guarantee that each marker is viewed at least by two cameras. Four markers were placed on the pelvis, one rigid cluster of three markers on each thigh and shank and three markers on each foot, giving a total of 22 markers attached to the body. After a successful tracking of the markers, the reconstructed 3D trajectories were used to calculate the angles of the joints. An example of the knee joint angles is shown in Fig. 2.14.

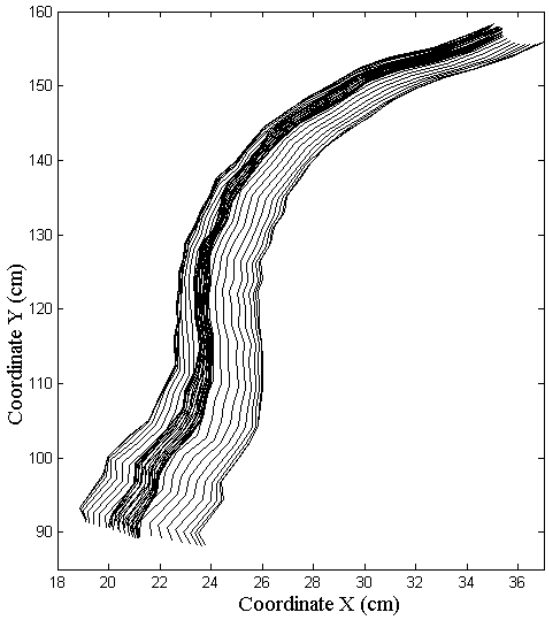


Fig. 2.12: Stick figures of spine curvatures during walking on a treadmill.

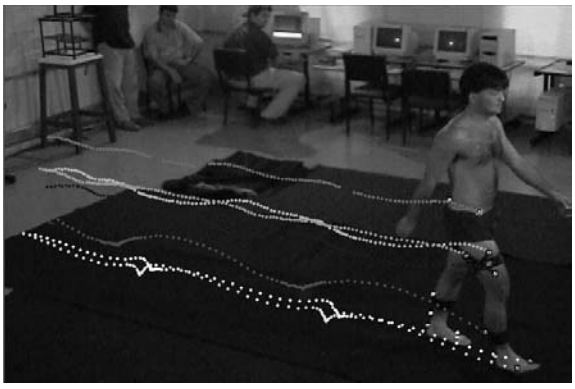


Fig. 2.13: One camera view tracking of reflexive markers applied for gait analysis.

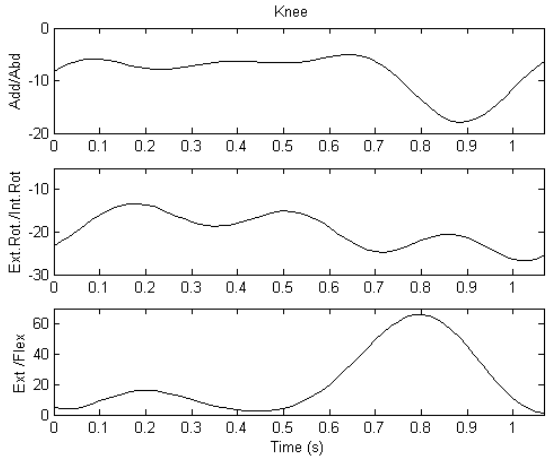


Fig. 2.14: Abduction / adduction (top), internal / external rotation (middle) and flexion / extension (bottom) knee angle calculated from the tracked markers.

For a Pentium III 700 with 128 Mbytes RAM, the execution rate of the frames, for the first example, was about 0.5 frames/s with a selected image region of 530x100 pixels. For this application, it is also possible to use a simple segmentation algorithm based on the intensity of the markers and a filtering by thresholds. In such a case, we can reduce the execution time to 2.5 frames/s in spite of a lower precision and an algorithm more sensitive to interference.

In the second application, by considering a search region of 21x21 pixels, the frame execution rate was about 9.5 frames/s for each marker. Since most of the time is taken to load the image, tracking multiple markers at the same time reduces the average execution time for each marker. In case of occlusions, the tracking process stops and continues only when the corresponding marker becomes visible again.

2.4 Hardware and software specifications

This software was developed using the Microsoft Visual C++ version 6.0. Therefore, it runs in any standard PC with operational system Windows 95/98/2000/XP. There is no requirement concerned with hardware configuration but it is recommended to use large memory and high-speed processor in order to decrease the processing time. Additional information about this system is available from the corresponding authors.

2.5 Conclusions

In this work, we described a software for the tracking of markers used for human motion analysis. One of the main advantages of this software is its flexibility to build tracking algorithms in a non-supervised environment, depending on the markers features and the performed movement. Using only filtering by thresholds in the markers extraction, as it is the case for many optoelectronic systems, yields a system more sensitive to interferences such as light intensity, reflections, background, etc. Our aim here was to improve the segmentation step for the tracking process, thus increasing the robustness and the flexibility of the overall tracking system. Instead of restricting the experimental environment to solve the tracking problem, the software proposed in this work allows the definition of well-adapted algorithms to perform this task, giving the possibility of manual corrections, when needed. This leads to the development of low-cost video-based systems to be used in a wide range of applications concerned with human motion analysis.

Acknowledgement

This work was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

.

Capítulo 3

Algoritmo de extração do fundo para ambiente com mudanças de iluminação

Neste Capítulo abordamos o problema de mudanças de iluminação na extração de uma imagem de fundo em uma seqüência de vídeo. O artigo apresenta um algoritmo morfológico que leva em consideração a intensidade de cada pixel ao longo de vários quadros e elimina os objetos em movimento.

O problema abordado considera as grandes mudanças de iluminação que ocorrem principalmente em ambientes externos não controlados na presença, por exemplo, de sol e nuvens que podem provocar uma rápida alteração na intensidade das imagens. Mudanças de iluminação na segmentação dos objetos têm sido tratadas a partir de métodos estatísticos. Estes métodos classificam os pixels como parte do fundo ou do objeto em movimento a partir da informação do quadro atual e de parâmetros estatísticos como média e desvio padrão das mudanças. Estes métodos consideram, assim, apenas mudanças suaves de iluminação. No caso de grandes mudanças os pixels podem ser erroneamente classificados alterando, desta forma, o modelo do quadro do fundo e, conseqüentemente, a segmentação dos objetos em geral.

Neste trabalho consideramos vários quadros de uma mesma seqüência para se extrair os quadros do fundo, obtendo assim, uma informação mais global das mudanças. Nosso objetivo resume-se em nivelar o sinal 1D de um pixel, ao longo dos quadros, de forma a suprimir os vales e picos do sinal. Este vales e picos representam a presença dos objetos em movimento em um determinado pixel.

O método divide-se em três etapas: inicialmente, são aplicados uma abertura e um fechamento morfológicos para suavizar o sinal e eliminar picos e vales de pequena dimensão. Estas situações acontecem quando os objetos se movimentam em grande ve-

locidade. Os picos ou vales de maior largura gerados por objetos parados ou com pequena movimentação são eliminados utilizando a informação da dinâmica dos mínimos e máximos regionais. Finalmente, as situações específicas em que o objeto permanece parado nos extremos do trecho selecionado são consideradas através da média das mudanças da região formada pelos pixels numa determinada vizinhança.

O método foi aplicado para a segmetação de jogadores em jogos do campeonato brasileiro filmado com quatro câmeras durante o dia, na presença de sol e nuvens. Os resultados obtidos para as quatro seqüências de vídeo mostraram-se eficientes, apesar das grandes mudanças na intensidade das imagens. Finalmente, as regiões da imagem (blobs), representando os objetos de interesse e extraídas durante a segmentação, são armazenadas num arquivo para serem utilizadas no rastreamento dos jogadores, como descrito no capítulo 4.

Background recovering in outdoor image sequences: an example of soccer players segmentation[1]

P.J. Figueroa N. J. Leite

Institute of Computing
State University of Campinas
{pascual,neucimar}@ic.unicamp.br

R. M. Barros

Faculty of Physical Education
State University of Campinas
ricardo@fef.unicamp.br

Abstract

In this work, we consider the problem of background pixels information recovering which can be used, for example, in applications concerning segmentation and tracking of components in video images. Shortly, to recover the background of image sequences representing outdoor scenes, we consider a nonparametric morphological leveling operation which takes into account the specific problem of lighting changes and the fact that we can have both slow and fast motion in the scene. We illustrate the segmentation of players based on the difference between image sequences and the corresponding recovered background representation. We also discuss the reduction of shadows in digital video of soccer games and show the good results of the whole background recovering and segmentation process.

3.1 Introduction

Image segmentation is an important step in many computer vision applications including, for example, tracking of moving objects, detection of intrudes in surveillance systems and recognition of people activities. For this reason, the improvement of algorithms for automatic segmentation has demanded a great attention of the researchers during these last years. Many aspects can make the segmentation of the interest components of an image very difficult, particularly when the events happen in an uncontrolled environment, as it is the case for sport games in outdoor scenes. In such environments, usually, we have considerable changes of illumination on a sunny day, specially in case of a cloudy sky. Beside these aspects, we also have to consider, the size and shape changes of the moving objects and their shadows, as well as the shadows of other objects projected onto the scene which make more complex the whole segmentation process. For example, during a soccer game in the afternoon, we can see the shadows of the stadium structure moving on the playing field, together with the shadows of the players.

[1] artigo submetido à revista Image and Vision Computing

Background substraction is a simple and very common method used in the segmentation of moving objects. It consists in extracting a background image and making the difference between this background model and the current image [19]. In order to consider environmental changes, such as illumination, shadows and background objects, the background image needs to be regularly updated. For this purpose, some statistical adaptive methods have been proposed [47, 13, 27, 65, 31, 19]. These methods update the background by modeling each pixel as a Gaussian distribution and work well in scenes with moving objects and slow lighting changes. François and Medioni [27] modeled the background pixels as multi-dimensional Gaussian distributions in HSV color space. McKenna et. al. [47] also considered the Gaussian distribution for modeling the background pixels in RGB channels. They proposed a method which combines the value of the RGB pixels and the chromaticity values with local image gradient. Stauffer and Grimson [65] model each pixel as a mixture of multiple Gaussians in order to consider multiple motions. Rider et. al. [59] modeled each pixel based on Kalman filter. In all these methods, changes of illumination are assumed to occur slowly relative to the object motion. The background model is updated recursively using mean and variance information and the latest defined measurements.

In some sport games, the players can stand still for a long time and start moving, suddenly, with high velocity. In such a case, the use of statistical adaptive methods can yield players to be associated to the background extraction model. Besides, under abrupt lighting changes, it is very difficult to recover the background information after a player stop moving.

Haritaoglu et. al. [31] developed a system called W^4 in which the background is modeled using several frames and a Gaussian bimodal distribution. Each pixel is represented by three values: its minimum and maximum intensity values and the maximum difference between consecutive frames. They also consider slow changes of lighting and report that, in case of sudden changes of background illumination, such as clouds blocking the sun, the segmentation fails.

Under a change of illumination, it is difficult to determine the cause of the pixel intensity changes, from the information conveyed by one single frame of an image sequence, and even more difficult to keep a still object detected for a long time.

The method introduced here considers a recovering of the background image by using morphological operations [63, 30] such as opening, closing and dynamic [29]. As in [31], we consider the information from consecutive frames, but we split the whole image into small regions to take into account the regional lighting changes. For each pixel, we define a function representing the changes in their intensity value, along the selected frames, and morphological operators are applied in this function. The simple opening and closing morphological filtering eliminate the moving objects, while the dynamic notion is used to

define a nonparametric leveling operation which eliminates slow moving or long time still objects.

This paper is organized as follows. The next section introduces the basic morphological operations used in our algorithm. Section 3 describes the background recovering method, and section 4 illustrates a sequence of operations for soccer players extraction and shadows elimination. Finally, section 5 discusses some results of the application and conclusions are drawn in section 6.

3.2 Basic morphological operations

In this work, the proposed segmentation algorithm is based mainly on mathematical morphological transformations [63, 30] represented by the following basic operations:

Dilation and erosion: These two operations constitute the basis of the more complex morphological transformations and can be defined as follows. Let f and b be a function representing the image as a topographic surface and a structuring element, respectively, so that $f : F \to Z$, $b : G \to Z$ and $F, G \subset Z^2$.
The dilation of f by a flat structuring element (a small image whose height is zero), b is given by

$$f(x) \oplus b = \max_{z \in G,\, x-z \in F} \{f(x-z)\}. \tag{3.1}$$

and the erosion of f by the same flat structuring element b is

$$f(x) \ominus b = \min_{z \in G,\, x+z \in F} \{f(x+z)\}. \tag{3.2}$$

Opening and closing: By iteratively applying dilation and erosion, one can eliminate details of an image, which are smaller than the structuring element b, without affecting its global geometric features. Visually, opening smoothes contours by breaking narrow isthmuses and eliminating small islands of an image. It is defined as

$$f(x) \circ b = (f(x) \ominus b) \oplus b \tag{3.3}$$

On the other hand, closing smoothes contours by fills narrow gulfs and eliminating small holes. This operation is defined as

$$f(x) \bullet b = (f(x) \oplus b) \ominus b \tag{3.4}$$

Watershed: This transformation is a very important morphological operation used in image segmentation. An intuitive definition of the watershed function [10] is given by considering an image f as a topographic surface. This surface presents regional minima, as shown in Fig. 3.1. Suppose these regional minima are pierced and the surface is

submerged progressively into water, so that the level is the same at the entire surface. During this flooding process, dikes are built in order to avoid the join of water coming from two different catchment basins. At the end of flooding, the lines formed by the set of dikes constitute the watersheds of image f, and the regions defined by these dikes correspond to the influence zone of each regional minimum.

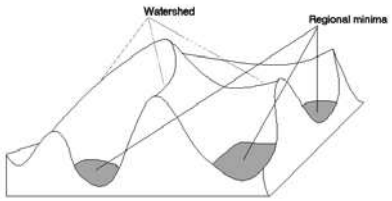


Fig. 3.1: An intuitive definition of watershed.

Dynamic of regional maxima: Dynamic is a measure of contrast of the image structures [29] . For a regional maximum M (Fig. 3.2), this measure is given by

$$dyn(M) = f(M) - \max\{\min\{f(x), x \in P\}\}, \tag{3.5}$$

where
$P = \{(p_0, p_1, ..., p_n)\},\ p_0 \in M \wedge f(p_n) > f(p_0)$

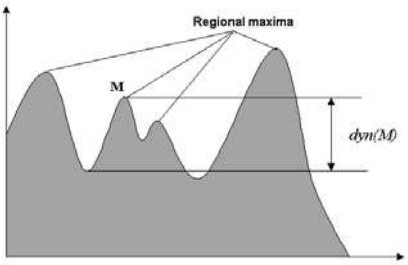


Fig. 3.2: Definition of dynamic for the regional maximum M.

Informally, the dynamic is defined as the minimal unevenness obtained in the path starting from a point M until another regional maximum higher than M is reached. The dynamic of the regional minima can be easily defined, in an analogous way, by considering the complement of the original image.

In the next section, we introduce a general algorithm for the recovering of the background information of video images based, mainly, on the above morphological operators.

3.3 Background recovering

As stated before, the unconstrained changing of illumination that occurs in outdoor scenes can make very difficult the segmentation of the interest components. In most cases, the intensity changes of the image pixels are not uniform, which means that they can be more significant in some regions than in others or that an increasing of intensity can affect some parts of the image while other ones become darker. This is a common case in temporary occlusions of the sun by clouds. Fig. 3.3 shows an example of these intensity changes for two different regions of the image in different frames of a soccer game video sequence. In the first region, the shadow of the goal disappears after 90 frames (at a rate of 7.5 fps), as a result of clouds blocking the sun, while, at the same time, in the other region of the field the shadows continue but the intensity of the images changes significantly. Fig. 3.4 illustrates the changes of intensity values in two pixels of the above two regions (Fig. 3.3), along 100 frames (about 13 seconds), and Fig. 3.5 shows the different intensity changes of two pixels of the same region, along the sequence, caused by reflection and nonuniform lighting.

In this work, to consider regional changes of illumination, we split the whole image into small regions, r, of size KxL, and propose a morphological algorithm for background recovering and, consequently, a segmentation method based on background subtraction.

Our background recovering algorithm uses N consecutive frames of the sequence and try to recover each background pixel by eliminating the moving objects through a morphological leveling operation. The algorithm consists, mainly, of the three steps described in the next subsections. The first step considers a simple filtering operation defined by a combination of morphological closing and opening. The second step uses the dynamic information of the signal to define a nonparametric leveling process which recovers the background pixel information even when the objects move slowly or stand still for a long time. Finally, the last step considers a specific case of the leveling process to extract the background pixels of the scene which were not totally recovered by the first two steps.

In this work, the morphological operations previously described are applied on one-dimensional signals representing the temporal information of a pixel x in frame k. Indeed, as we will see next, instead of an image $f(x)$, we consider operations on a discrete function $f_x(k)$. The intensity of this image is obtained by converting the RGB color image into a grayscale information.

3.3.1 Background recovering by closing-opening filtering

Let $f_x(k)$ be a function representing the intensities of one pixel $x \in Z^2$ along N frames ($k = 1..N$).
First, to smooth the original signal f, we apply an alternate sequential filter given simply

Fig. 3.3: Two image regions in different frames of a soccer game showing changes of illumination.

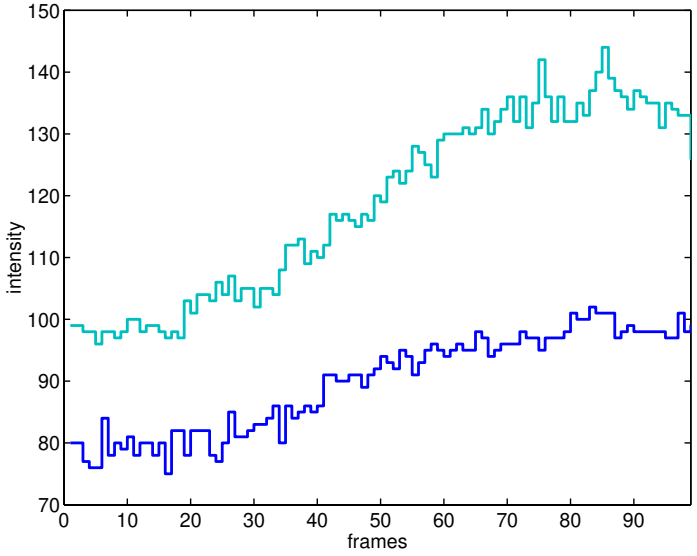


Fig. 3.4: Intensity changes of pixels in two different image regions caused by a change of illumination in an outdoor scene.

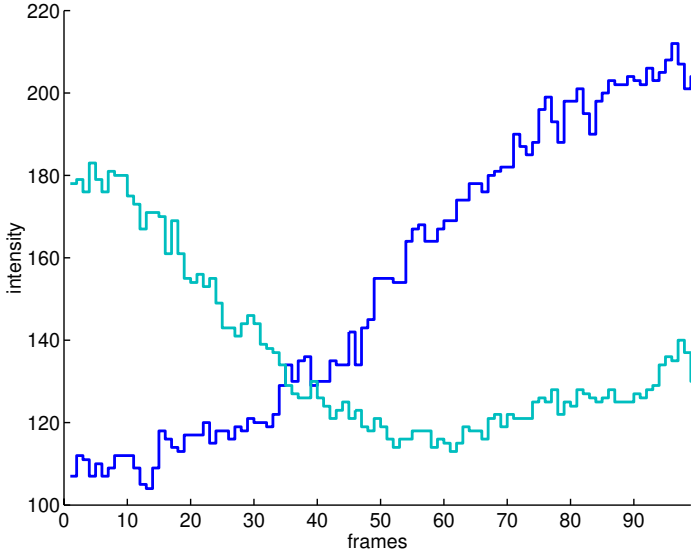


Fig. 3.5: Intensity changes of two pixels caused by reflection and nonuniform illumination.

by a closing followed by an opening morphological operation with a symmetric structuring element of size 1x5. The smoothed function, $g_x(k)$, is

$$g_x(k) = (f_x(k) \bullet b) \circ b \tag{3.6}$$

As we can see in Fig. 3.6, this operation eliminates narrow peaks and valleys related to the moving objects of the sequence (the significant peaks and valleys), yielding a partial recovering of the background information along this sequence.

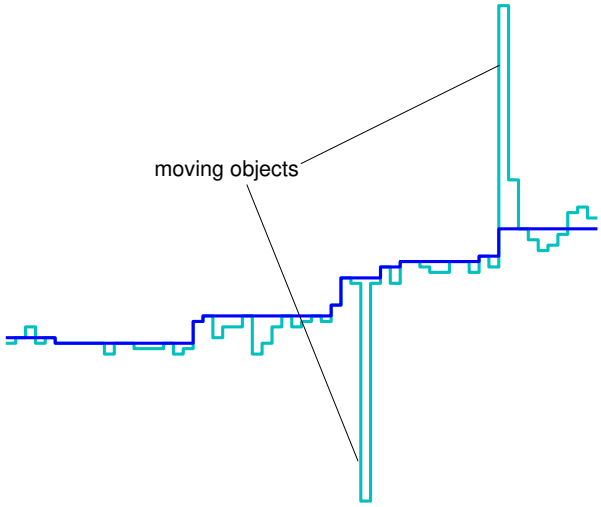


Fig. 3.6: Result of a temporal closing-opening operation with a 1x5 symmetric structuring element applied to one pixel along 60 frames.

Most moving objects can be detected by the above smoothing operation which completely filters out valleys and peaks of the signal depending on the size of the used structuring element (Fig. 3.6). However, there are cases in which one object may stand still for a while (e.g., more than 5 frames) or move very slowly. In such cases, we need a more complex additional procedure to eliminate these components. Fig. 3.7 shows the intensity changes of a pixel, along 60 frames, in which we can see the presence of the darker slow moving objects, represented by low-level signals, between high-level components representing the background signal. By applying the morphological closing-opening operation on the signal in Fig. 3.7, we obtained the result shown in Fig. 3.8. In this figure, we can still see the presence of valleys in the signal.

3.3.2 Background recovering using dynamic information

The main problem of the background recovering concerns the classification of the pixels as belonging to the background or to a moving object, specially when there are significant

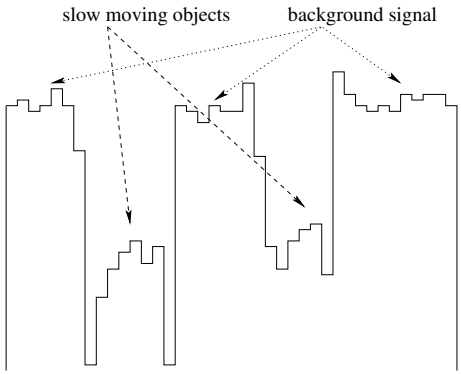


Fig. 3.7: Original function.

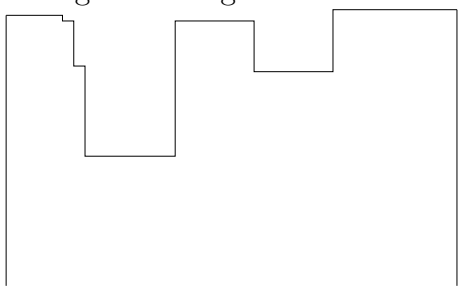


Fig. 3.8: The function in Fig. 3.7 after a closing-opening smoothing.

changes in the background caused by different lighting conditions. Although the gray-scale variation of the background pixels is expected to be smooth, the environmental reflection process can generate abrupt changes in the image intensity.

Let μ_x and σ_x be, respectively, the mean and standard deviation of the intensity difference of pixel x between two consecutive frames along N frames. Also, let r be a small KxL region of the original image, with mean μ_r and standard desviation σ_r calculated as follows:

$$\mu_r = \frac{1}{K * L} \sum_{x \in r} \mu_x \tag{3.7}$$

$$\sigma_r = \frac{1}{K * L} \sum_{x \in r} \sigma_x \tag{3.8}$$

In order to define the pixel x along the sequence as belonging only to background or to some part of the moving objects, we consider two gradient values from the smoothed function g_x (Eq. 3.6), namely, the sum of gradients S_Γ and the sum of the module of the gradients $S_{A\Gamma}$, for every frame k where the module of the gradient is greater than some threshold T_r. The value of T_r is defined by considering the intensity changes of the region r containing x, and is based on the mean value μ_r and the standard deviation σ_r of this

region. These are defined as follows:

$$T_r = \mu_r + 2\sigma_r \ , \tag{3.9}$$

$$S_\Gamma = \sum_{k=2}^{N} d_x(k), \quad if \quad |d_x(k)| > T_r \ , \tag{3.10}$$

$$S_{A\Gamma} = \sum_{k=2}^{N} |d_x(k)|, \quad if \quad |d_x(k)| > T_r \ , \tag{3.11}$$

where

$$d_x(k) = g_x(k) - g_x(k-1), \ \ k = 2, ..., N. \tag{3.12}$$

If $S_{A\Gamma} = 0$, i.e., no value of k satisfies the condition in Eq. 3.11, we consider the pixel x as a background pixel along the N processed frames. $S_{A\Gamma} > T_r$ means high uneven values in the function and, consequently, we may detect the presence of high peaks and/or deep valleys by the following relation

$$S = S_{A\Gamma} - |S_\Gamma| \tag{3.13}$$

Let S_u and S_d be the absolute value of the sum of positive ($d_x(k) > Tr$) and negative ($d_x(k) < -Tr$) gradients, respectively. In this case, we have

$$S = S_u + S_d - |S_u - S_d| \tag{3.14}$$

If $S_u > S_d$ then $S = 2S_d$, otherwise $S = 2S_u$. To guarantee the existence of peaks or valleys in the signal (Fig. 3.9), S_u and S_d must be different of zero and greater than T_r, yielding

$$S > 2T_r. \tag{3.15}$$

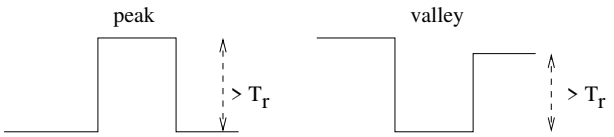


Fig. 3.9: Example of peak and valley configurations.

As we have seen before, the significant peaks and/or valleys of the signal indicate the presence of slow moving or still objects (see Fig. 3.7). To recover the background by suppressing these image extrema, we define a leveling process based on the dynamics of the regional maxima and/or minima of the signal for the pixels satisfying condition in Eq. 3.15. This background recovering does not depend on any structuring element information and, as we will see next, it needs only two steps for the complete suppression of the significant peaks and valleys of the images.

Since the dynamic computation depends on the type of the signal extrema to be processed (peaks or valleys), we need to identify a priori each one of these types. For this purpose, we consider the summation of positive, $S_{\Gamma P}$, and negative, $S_{\Gamma N}$, differences with respect to the first value $g_x(1)$.

$$S_{\Gamma P} = \sum_{k=2}^{N} d'_x(k), \qquad \text{if} \quad d'_x(k) > T_r \tag{3.16}$$

$$S_{\Gamma N} = \sum_{k=2}^{N} d'_x(k), \qquad \text{if} \quad d'_x(k) < -T_r \tag{3.17}$$

where $d'_x(k) = g_x(k) - g_x(1)$, $k = 2...N$.

From the above equations, we consider that there are significant peaks in the smoothed signal g if $S_{\Gamma P} > T_r$ and, consequently, we compute the dynamic of its regional maxima. In a similar way, we consider that there are significant valleys in the signal if $|S_{\Gamma N}| > T_r$ and, in such a case, we compute the dynamic of its regional minima. If both significant peaks and valleys are present in the smoothed signal g, then we compute the dynamics of its regional maxima and minima. Note that the consideration of these specific cases are important only to define a less time-consuming method since, in a general way, we can always compute the dynamics of both regional maxima and minima of g regardless the above differences information.

The dynamic of the regional minima, as explained in section 3.2, is computed here by considering the notion of the flooding process of the catchment basins for the determination of the watershed lines (Fig. 3.10). From this process, each regional minimum and its influence zone is labeled, as shown in Fig. 3.10. By considering the dynamic information, we define a new function, w_x, which will be used to fill in the valleys and suppress the peaks of the smoothed signal g_x, independently of their size.

Let $D_x(l)$ and $F_x(l)$ be the dynamic value of a regional minimum l and the value of the points in this regional minimum, respectively. Also, let $L_x(k)$ be the label of the signal in frame k representing the influence zone of the corresponding regional minimum. In order to guarantee a complete filling in of the valleys, we let merge different regional minima of the signal, yielding new labels $L'_x(k)$ for the merged points (Fig. 3.11). This process starts with the lowest level regions and is executed while $g_x(k) \leq F_x(L_x(k)) + D_x(L_x(k))$.

The function w_x, which levels the peaks and valleys of the smoothed signal g, can be defined as follows

$$w_x(k) = D_x(L'_x(k)) + F_x(L'_x(k)) - g_x(k), \qquad k = 1, ..., N \tag{3.18}$$

As we can see in Fig. 3.12, this function indicates the values we need to add to the original signal to fill in its gaps representing regional minima.

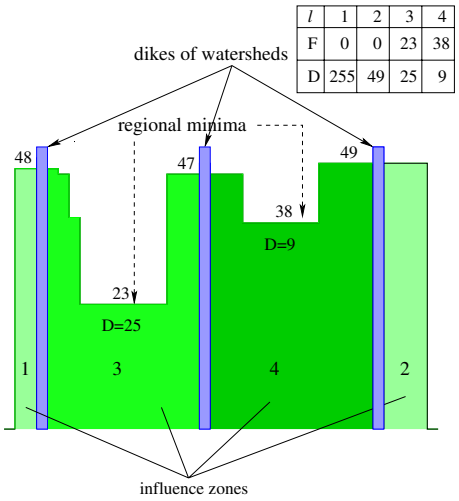


Fig. 3.10: Dynamic computation.

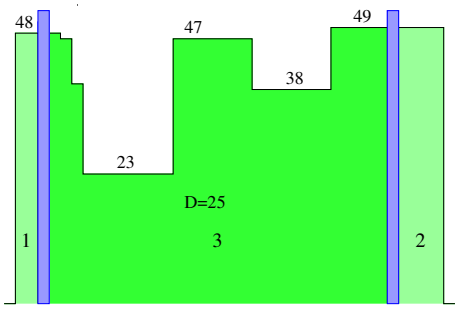


Fig. 3.11: New labeling of the influence zones in Fig. 3.10.

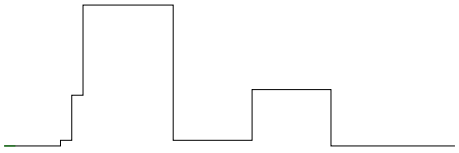


Fig. 3.12: Value of the function used to fill in the valleys in Fig. 3.11.

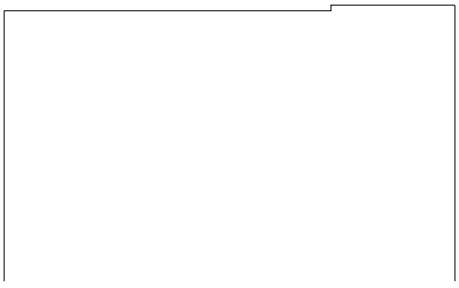


Fig. 3.13: The recovered background function.

The dynamic of the regional maxima is calculated in a similar way by considering the complement of function g_x. In such a case, to level the image peaks, we subtract function w_x from the original signal. Thus, from the dynamic information, we can define a function h_x, which fill in valleys and suppress peaks of a signal, as follows

$$h_x = \begin{cases} g_x + w_x, & \text{in case of valleys} \\ g_x - w_x, & \text{in case of peaks} \end{cases} \tag{3.19}$$

The result of the leveling operation of the signal in Fig. 3.10, according to Eq. 3.19, is shown in Fig. 3.13. Fig. 3.14 shows an original signal (Fig. 3.14.a), containing both valleys and peaks, followed by a closing-opening morphological filtering (Fig. 3.14.b). Fig. 3.14.c and 3.14.d show, respectively, the corresponding w_x function in case of valleys and peaks. Finally, Fig. 3.14.e shows the result of function h_x (Eq. 3.19) which, in our application, will be used to recover background pixels of sport scenes, as we will illustrate elsewhere.

Other examples of background pixel recovering, in the case of significant illumination changes, are shown in Fig. 3.15 and Fig. 3.16. The results of the nonparametric leveling process (the dark lines) show the good performance of the method.

3.3.3 Background construction using the regional mean

From the discussion in section 3.3.2, we can note that the above leveling process works well when we have a complete sequence of information represented here by a background followed by moving objects which, in turn, are followed by a background. In such a case, the moving objets constitute the peaks or valleys of the one-dimensional processed signal.

In this section, we consider the specific problem in which we do not have this complete signal representation, as illustrated in Fig. 3.17, where the high uneven part of the signal (after frame 90) represents a moving object which is not followed by a background information. Indeed, the above method for background pixel recovering eliminates all the moving objects if they leave their position before the end of the N processed frames. In case of still or slow moving objects, during the last set of processed frames, we do not have the desired configuration of valleys or peaks and, consequently, we cannot properly recover the background pixels at this position of the signal.

A general but time-consuming way to solve this problem is to consider a variable number of frames during the processing, such that the expected configuration given by a sequence of background, moving objects, and background is always defined.

In this subsection, we propose to solve this specific problem by a simple leveling operation which takes into account the regional mean of the changes in the images. Here, we process only pixels with high uneven values, i.e., when $|S_\Gamma| > T_r$.

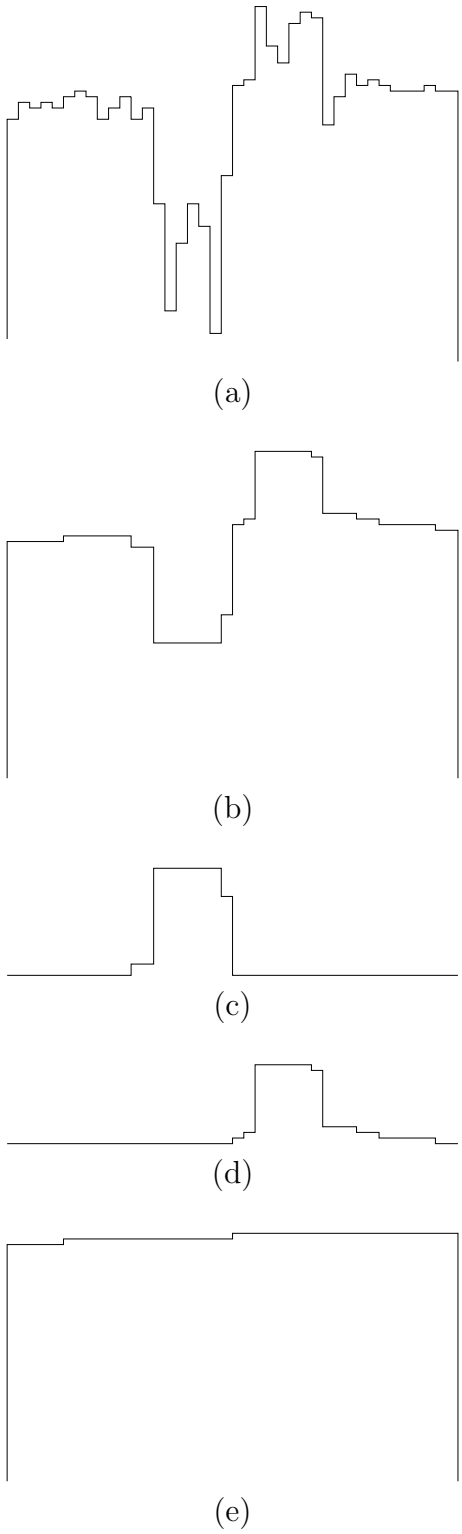


Fig. 3.14: Example of background pixel recovering. (a) Original function of one pixel in the sequence. (b) Smoothing by closing-opening operation. (c) The function to fill in valleys of the smoothed signal. (d) The function to suppress the peaks of the smoothed signal. (e) The result of the background pixel recovering.

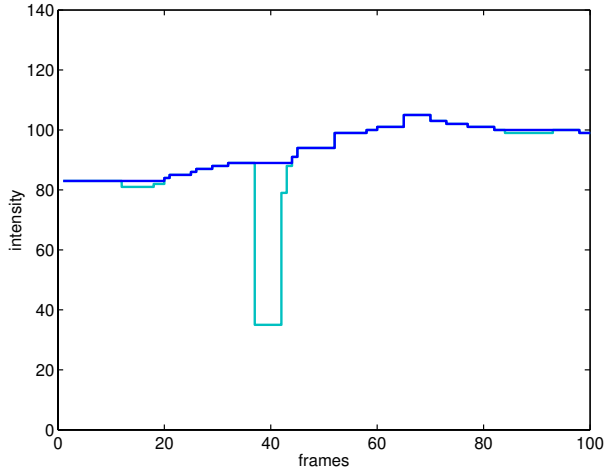


Fig. 3.15: Result of background pixel recovering by considering the dynamics of the regional minima.

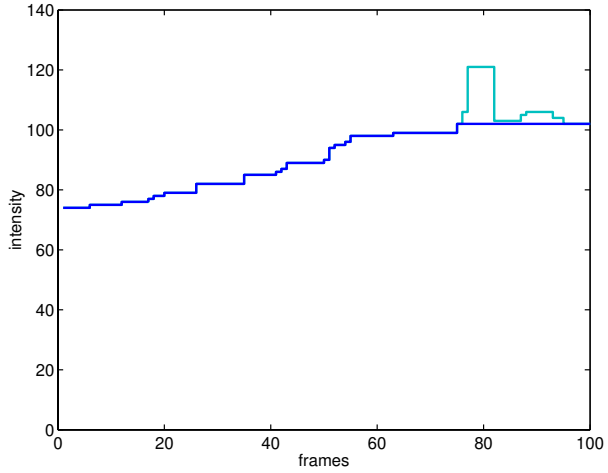


Fig. 3.16: Result of background pixel recovering by considering the dynamics of the regional maxima.

In such a case, the background pixels are defined based on the accumulative sum of the mean of the gradient, d_y, of the background pixels belonging to the small regions r. For each background pixel $y \in r$, we compute the gradient $d_y(k)$ in each frame $k = 2...N$. Let k' be the frame corresponding to the first detected uneven point such that $|d_y(k)| > T_r$. A regional mean of gradients d_y, in a certain region r, is defined as follows:

$$m_r(k) = \frac{1}{n_r} \sum_{y \in r} d_y(k), \quad k = k'..N \tag{3.20}$$

where n_r is the number of background pixels, i.e., those points from which $S_{A\Gamma} = 0$, according to section 3.3.2.

The accumulative sum of mean, A_r, is given by

$$A_r(k) = \sum_{i=k'}^{k} m_r(i), \quad k = k'..N \tag{3.21}$$

Finally, the occluded pixel is defined as

$$h_x(k) = g_x(k' - 1) + A_r(k), \forall x \in r \quad k = k'..N \tag{3.22}$$

We consider that the frame at location $k' - 1$ is a background pixel. An example of background recovering using this approach is shown in Fig. 3.18.

In Fig. 3.19, we show a pixel color map where each color represents the method used for the background pixel recovering for a sequence of 100 frames. The white regions correspond to the original background of the sequence while the light gray ones represent the background pixels recovered by the smoothing operator only (section 3.3.1). The dark grey regions show the background pixels after the leveling process based on dynamics. The pixels recovered by the regional mean of the image regions are shown with black color.

3.4 The soccer players segmentation

After the background recovering, the extraction of the players in a soccer game is performed by a set of simple operations consisting of:

- The difference between the current frame and the recovered playing field image.
- A simple morphological filtering (opening and closing) to eliminate noise.
- A binarization of the filtered image by a thresholding operation taking as threshold the value T_r previously defined (Sec. 3.3.2).

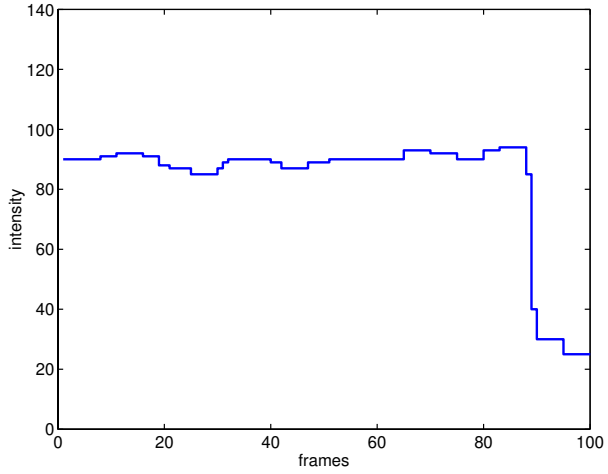


Fig. 3.17: An example of unevenness configuration.

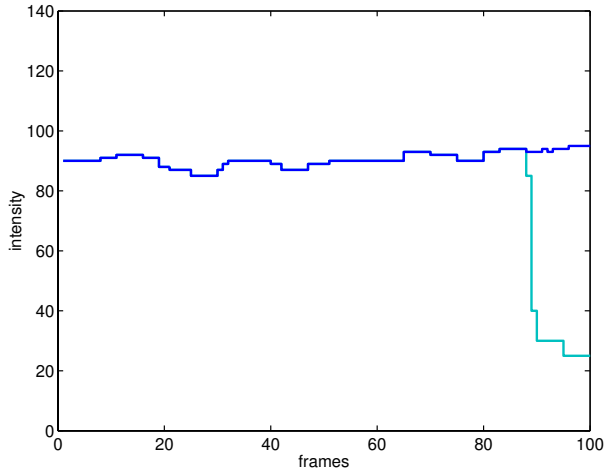


Fig. 3.18: Background recovering of the uneven points in Fig. 3.17 using regional mean information.

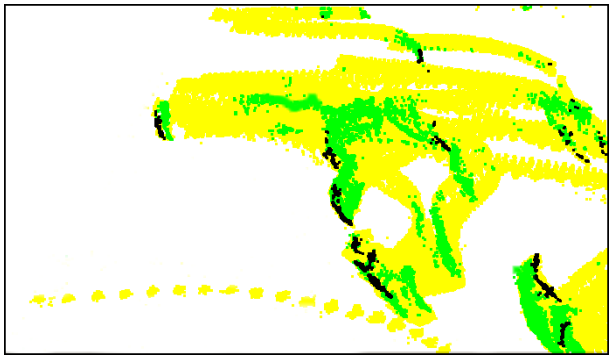


Fig. 3.19: Pixel map showing the methods used for the background pixel recovering.

Fig. 3.20: An image from a soccer game (frame 14525).

- A labeling of connected pixels and extraction of the regions (*blobs*) representing the moving objects.

Figs. 3.20-3.22 illustrate the segmentation problems under different lighting conditions. Note that for a range of only 75 frames (from frame 14525 in Fig. 3.20 to frame 14600 in Fig.3.21), the shadows of the soccer players appeared in the scene (Fig. 3.21), making dificult the whole temporal segmentation process. The final result of our segmentation algorithm, for the image in Fig. 3.21, is shown in Fig. 3.22. Note that the method was restricted to the playing field region. This result shows that all the soccer players in the image were correctly segmented, including their shadows.

Since for some applications (e.g., tracking of the players along the sequence) these shadows can constitute undesirable information, we add to the method simple morphological transformations to eliminate them, based on some specific geometrical features.

As we can see in the original image (Fig. 3.21), the shadows of the players have less information (in terms of number of pixels) in the vertical direction than the soccer players. At this point, all the processing is applied only to each blob defined by the above

Fig. 3.21: An image from a soccer game (frame 14600).

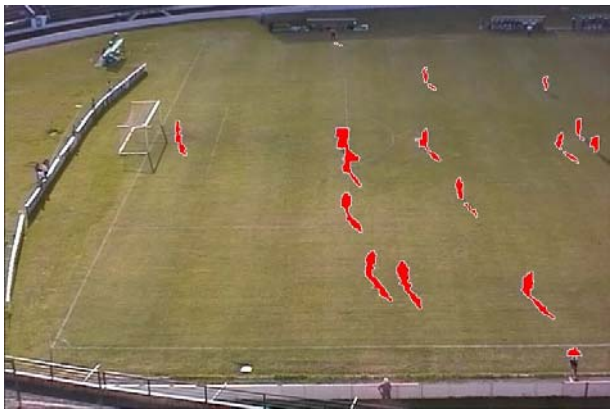


Fig. 3.22: Segmentation of the player.

Fig. 3.23: Elimination of shadows by simple morphological operations.

segmentation algorithm. To suppress the shadows of the moving objects, the following set of morphological binary operations is applied to the segmented image

1. Opening of the segmented image with a structuring element of size $(2n + 1)$x1, where n is proportional to the size of the players defined during the calibration of the camera. Here, we cannot simply use the blobs size since one blob may represent more than one soccer player in the image.

2. A conditional dilation (limited to the original blob regions) of the open set defined above, $n/2$ times, with a simetric structuring element of size 1x3. This dilation aims at recovering parts of the players eliminated by the opening operation with the vertical structuring element $(2n + 1)$x1.

Fig. 3.24 illustrates this operation for a blob in Fig. 3.22. Fig. 3.23 shows the results of the shadows elimination algorithm for all the blobs in Fig. 3.22.

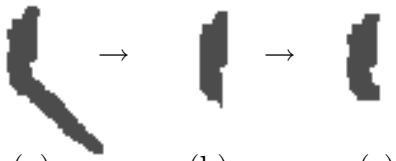


Fig. 3.24: A set of simple operators for shadows elimination. (a)Original segmented blob. (b)The blob after a morphological opening. (c)The result after a conditional morphological dilation.

3.5 Applications and some results

We tested the proposed algorithm in sequences of soccer games filmed with four cameras on a sunny and cloudy day. The tests were executed on a Pentium III 700 MHz PC with 512 MB of memory, at a rate of 7.5 fps. The results obtained were satisfactory and despite the significant changes of illumination the soccer players were correctly segmented. The data in Tables 3.1-3.2 were obtained for a 10 minute processing (4500 frames) of the four video images. Table 3.1 shows the total number of extracted blobs and the number of false and lost blobs. Most of the lost components are related to the linesmen whose blobs size can be very small since they are located on the extrema of the segmentation region. It is interesting to note that during the game the goalkeepers stood still for a long time (e.g., more than one minute), and even in such a case, they were successfully segmented. Some of the false defined blobs are related to the error in the initial frames or to erroneous detection of unevenness of the signal.

The size of the processed image and the execution times are given in the table 3.2. Note that the size of the images is different for each camera because we process only

Sequence	total blobs	false blobs (noise)	lost blobs
Camera 1	23119	27 (0.12%)	130(0.56%)
Camera 2	50773	22 (0.04%)	10(0.02%)
Camera 3	61315	98 (0.16%)	66(0.11%)
Camera 4	35707	59 (0.17%)	77(0.22%)

Table 3.1: Result of blobs segmentation in a soccer game.

Sequence	image size	processing time (min)	processing rate (frame/s)
Camera 1	687x397	51	1.47
Camera 2	710x398	56	1.34
Camera 3	706x418	57	1.32
Camera 4	693x388	52	1.44

Table 3.2: Segmentation times for different image sequences.

the regions limited to the playing field. For the background recovering, we considered N=100 frames and small regions r of size 20x10. In case of slow changes of illumination, the processing time can be reduced by decreasing the frequency of the processed frames during the background recovering, in which one background image information is used for more than one frame. In such a way, the processing time can be reduced, reaching, thus, a real-time processing. Furthemore, the method can be easily parallelized, since the whole processing takes into account only regional information.

3.6 Conclusion

In this work, we considered the problem of background recovering in video images related to segmentation of soccer players. We defined a morphological nonparametric leveling operation which takes into account information related both with the background and the foreground of video images representing outdoor environment. We illustrated the proposed method on a sequence of a soccer game video and showed also how the background recovering algorithm can be used for segmenting the players and their shadows. Further, this segmentation step will be useful, for example, in the tracking of these players by a system that aims at recognizing events and analysing the performance of the players on a soccer playing field. These two aspects constitute our current work on this matter.

Capítulo 4

Rastreamento de jogadores de futebol

Este capítulo aborda o problema de rastreamento de jogadores de futebol visando a determinação da posição de todos os jogadores, em cada instante do tempo, durante uma partida. Para tanto, os jogos são filmados por no mínimo 4 câmeras posicionadas no estádio de tal forma a cobrir, em conjunto, o campo inteiro.

Foram desenvolvidos algoritmos de segmentação e rastreamento de jogadores baseados em técnicas de processamento de imagens e visão computacional. Os dados extraídos da seqüência de imagens são convertidos em dados representando a posição real dos jogadores no campo. Estes dados podem ser empregados na análise cinemática da movimentação dos jogadores, através do cálculo de distâncias e velocidades atingidas pelos mesmos em determinados intervalos de tempo.

A detecção automática de jogadores, a partir de imagens de vídeo, tem sido abordada através de técnicas de casamento de padrões [68, 73], informação de histogramas [73, 39], informações de cor [37, 70] ou métodos baseados em diferença de imagens [52, 45]. A maioria destes métodos tem-se mostrado eficiente na etapa de segmentação dos jogadores, com algum grau de precisão e dependendo das condições experimentais. No entanto, uma das principais dificuldades da aplicação, reside no rastreamento dos jogadores (para a obteção da sua posição real no campo, ao longo da seqüência), especialmente em casos de oclusões.

No futebol, as oclusões são muito freqüentes e qualquer método de rastreamento proposto deve tratar estas situações visando um rastreamento correto e uma boa precisão dos dados. Alguns métodos têm tratado oclusões a partir da projeção de histogramas [73], no caso de imagens com boa definição de cores. Outras abordagens consideram a trajetória dos jogadores em combinação com o Filtro de Kalman [52].

A segmentação neste trabalho é abordada considerando o método da diferencia de

imagens, ou seja, entre a imagem do quadro atual e uma imagem do modelo do fundo. Este tipo de segmentação é descrito no Capítulo 3. O processo de rastreamento dos jogadores é tratado usando uma representação sob a forma de grafos, em que, os nós representam os blobs obtidos na segmentação e as arestas, as distâncias entre blobs de dois quadros consecutivos. Através desta estrutura de dados, o rastreamento de um jogador resume-se na busca de caminhos mínimos a partir de um nó inicial do grafo. O problema das oclusões é considerado a partir de subdivisões dos blobs, utilizando operações morfológicas e informações adicionais sobre a forma, tons de cinza, etc., armazenadas no grafo.

O algoritmo foi aplicado no rastreamento de jogadores em jogos oficiais do campeonato brasileiro de futebol. Como resultado, apresentamos no artigo que segue a avaliação do método para uma seqüência de 10 minutos, em termos de número de oclusões corretamente resolvidas, número de intervenções manuais e número de quadros rastreados automaticamente.

O método proposto foi avaliado ainda em termos de precisão [49] e através de uma análise de 90 minutos de jogo [48].

Tracking soccer players using multiple cameras[1]

P.J. Figueroa N. J. Leite

Institute of Computing
State University of Campinas
{pascual,neucimar}@ic.unicamp.br

R. M. Barros

Faculty of Physical Education
State University of Campinas
ricardo@fef.unicamp.br

Abstract

In this work, we consider the problem of tracking players, during a soccer game, through the use of multiple cameras. The main goal here consists in finding the position of the players on the pitch at each instance of time. The occlusion is treated by splitting segmented blobs and by considering multiple views of overlapped image regions. The tracking is performed through a graph representation in which the nodes correspond to the blobs obtained by image segmentation and the edges, weighted using the blobs color and trajectory in the image sequence, represent the distance between nodes.

4.1 Introduction

Soccer is a very popular sport in the world and there is a great interest in better understanding its important fundaments in order to increase the performance of a team and better adapt the planning of the trainings. The movement of the players on the field, as a function of time, is a useful information that can contribute for improving the performance of the players at different positions [3]. For tactical variations that a team can assume during a game, for example, the measured values may be associated to physiological variables as well as to technical and tactical information [57, 7].

The first studies concerned with the players movement during the game were made by Reilly et al. [57] which employed audio recorders to register the estimated location of the players. Withers et al. [72] used a camera to analyze the movement of a unique soccer player. Mayhew and Wenger [46] also used a camera to track two players, each one filmed alternately for 7 minutes. They computed the time spent for each activity of these tracked players, such as walking, running, jogging, staying, as well as the frequency of the corresponding activities.

Aiming to better quantify the players movements, Erdman [20] filmed a soccer game with one stationary TV camera (using wide-angle lens of 130°) and analyzed the dis-

[1] artigo submetido à revista Computer Vision and Image Understanding

placement of one player, by replaying frame by frame, using a videotape player and a transparent squared sheet adapted to the monitor of the screen. The player position was annotated each one second, during five minutes, and the kinematic quantities were calculated.

Henning and Briehle [33] analyzed the soccer players movement by using a global positioning system (GPS). This kind of system locates the global position of the object by satellites which receive the signal emitted by a transmitter located on earth's surface. This methodology demands a device of 250 grams to be carried by the tracked object from which the data are collected at a frequency of 1 Hz.

D'Ottavio [17] presented a method based on a potentiometer and two cameras for tracking one player during 90 minutes. The operator focus on the player of interest and follows his movement. The information stored in the potentiometer allows to calculate the corresponding player position as a function of time.

Recent advances in video technology and computer processing performance have motivated the interest of researchers in using computer vision and image processing techniques for the automatic analysis of the sport games by videogrammetry.

One of the first works treating the problem of automatic tracking sport players was made by Intille and Bobick [35] which developed a technique called *closed-worlds* applied in the tracking of american football players. A *closed-world* is defined as a region of time and space in which the specific context is adequate to determine all possible objects present in that region, such as players, yard-lines, hash-marks, grass and so forth. The authors report that this method works better than the common template matching for tracking isolated players and that their main difficulty is to treat the case in which a *closed-world* representation contains more than one player.

Taki et al. [68] present a method for a quantitative evaluation of a team work in soccer games. For this purpose, they considered only static cameras and isolated players which are tracked using template matching. Seo et al. [73] use the TV broadcasted images and a tracking method based on template matchings and on the histogram backprojection concept to solve the occlusion problems. Neadham and Boyle [52] proposed a method named condensation for multi-object tracking. The method was applied for sequences of futsal video images. In [69], different systems of color representation are used in the segmentation of soccer players. Matsui et al. [45] also present a work for animation of soccer scenes using TV broadcasted images.

As it is the case for other tracking systems based on video cameras, the tracking methods need to solve different problems concerned with extraction, identification and correct trajectory definition of the players. Furthermore, the outdoor unconstrained environment and the very common occlusion problems make the correct tracking of all the players more difficult.

Indeed, one of the most challenging problems related to the tracking of soccer players concerns occlusion and the player congestion, which occur, especially, in cases of free kicks and corners. Iwase and Saito in [36] uses 8 cameras covering the whole region of the goal (the penalty box) in order to treat this problem. However, this kind of solution is expensive and the occlusion problems are not totally solved.

In this work, we propose a method for tracking soccer players based on at least 4 static cameras which together should cover the whole playing field. As we will see elsewhere, by considering a simple model of the players, and some morphological operations, we treat occlusion and congestion problems through a basic splitting of segmented blobs representing set of two or more partially occluded players.

The overlapped regions related to the four cameras are used for synchronization and to solve the many cases of occlusions. This kind of configuration, based on the placement of a set of cameras on the pitch, provides enough reference points necessary to the calibration of the cameras, and guarantees that the size of the players in the image is big enough to discriminate noise and the interested components of the scene.

The first step in our tracking algorithm concerns the segmentation of the players while a second one detects their correct trajectories. The segmentation step consists in extracting the blobs representing these players. Further, we construct a graph having the blobs as nodes and find the players trajectory by a minimal path searching on this graph. The edges of the graph are defined by considering information such as distance between blobs, colors (grey level intensity), and the movement direction of the tracked players. Here, we aim at isolating the players by splitting their corresponding blobs and correctly treating some cases of occlusions.

This paper is organized as follows: the next section introduces the segmentation algorithm. Sec. 3 describes the tracking procedure in details. Sec. 4 illustrates and discusses the results of the method when applied to real sequences of soccer images and, finally, some conclusions are drawn in Sec. 5.

4.2 Segmentation of the players

Background substraction is a simple and very common method used for segmenting moving objects which consists of the difference between a set of images and its background model. To consider problems in outdoor scenes such as changes of illumination, shadows, background objects etc, these methods need to frequently update the background representation model. For this purpose, some statistical adaptive methods [47, 13, 27, 65] have been discussed in the literature. These methods, which work well for relatively simple scenes with slow changes of illumination, update the background model for each frame by assuming its background as a Gaussian distribution. This approach can easily incorporate

to the model objects that stop moving for a certain time. In some applications, however, it is desirable to keep these still or slow moving objects correctly tracked. In soccer games, for example, it is common to have players that stand still for many video frames.

In cases of sudden changes of illumination and background objects, we need to consider more complex procedures taking into account some additional information of the analyzed sequence. For example, Haritaoglu et. al [31] model the background using several consecutive frames and the Gaussian bimodal distribution, where each pixel is represented by the minimum and maximum intensity values and the maximum difference between frames.

In this work, we extract the background image by applying a simple median filter along the pixels of some consecutive frames, thus, regularly updating the background image of a certain number of the video frames. The intensity of each background pixel is calculated as the mean of the most likely repeated intensities along the considered set of frames. This simple method correctly updates the background image for the case of slow changes of illumination and keep most of the still objects detected.

Our complete segmentation algorithm consists of the following basic steps:

- Background extraction.
- Difference between the current frame and the image corresponding to the extracted background.
- Morphological filtering (opening and closing) to eliminate noise.
- Image binarization by thresholding.
- Labeling of the connected pixels and definition of the corresponding regions as *blobs*.

Fig. 4.1 shows a frame of a soccer video sequence and Fig. 4.2 shows the corresponding updated background considering a set of 20 frames. Fig. 4.3 shows the final result of the segmentation step.

4.3 Tracking players

In this section, we consider the specific problem of tracking the blobs representing the players and, finally, the definition of their location on the playing field. As mentioned early, one of the main difficulties of the tracking process concerns the partial or total temporal occlusion of the objects. Hence, the splitting of the blobs aiming at separating or isolating the players is also considered in this section. This splitting takes into account the spatial-temporal information of the image sequence. The spatial information is explored

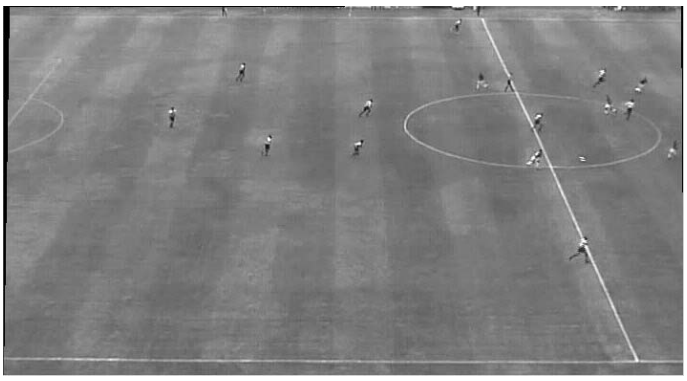


Fig. 4.1: A frame of a soccer video.

Fig. 4.2: The background image.

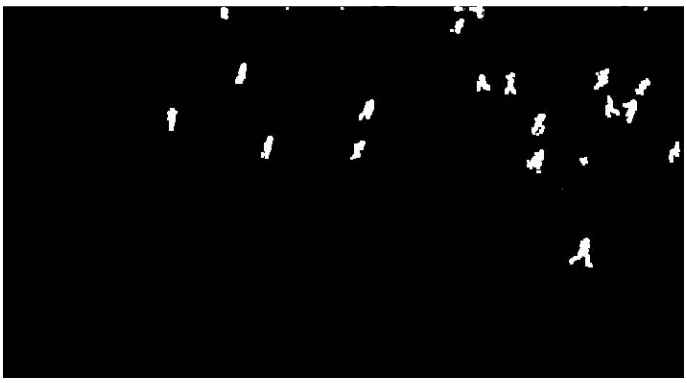


Fig. 4.3: The blobs in the segmented image.

by considering the size, shape and color of the blobs, while the temporal information explores the relation between blobs in different frames. In this work, we use a graph representation to define this temporal dependence.

The graph is constructed from the set of blobs obtained during the segmentation step in such a way that nodes represent blobs, and edges represent, the distance between these blobs. Therefore, each node of the graph stores the spatial information of a blob while an edge conveys the temporal information related to the dependence between blobs. This representation model allows us to better approach the correspondence problem of the objects which can help not only in the splitting of the blobs but also in correctly tracking them along the sequence. The tracking of each player is performed by a minimal path searching in the graph. Besides the distance between blobs, the edges of the graph are weighted by considering information such as velocity, orientation and color of the blobs.

Since the graph constitutes the data structure of the proposed method, the next subsection discusses its construction in details.

4.3.1 The graph construction

Let G be an oriented graph of a video sequence. Our data structure can be defined according to the following steps:

1. Creation of a node $n_{i,1}$ for each blob i in the first frame, $t = 1$, and insertion of this node into graph G.

2. Computation, for each blob i in frame t, of the distance $d_{i,j}$ between nodes i and j, in frame $t + 1$.

3. Creation of an edge (i, j) satisfying condition $d_{i,j} < d_{max}$, where d_{max} represents a given maximal distance.

4. Creation of a node $n_{j,t}$ for each blob j in frame t and insertion into G.

5. Repetition of steps 2-4 for the whole video sequence.

Fig. 4.4 shows some nodes and edges of the graph for four consecutive frames. Each node stores information about the blobs features defined during the graph construction. Examples of these information are:

- *width* and *height*: the size of the bounding rectangle of the blobs.
- *area*: the number of pixels of a blob.
- *perimeter*: the number of pixels in the contour of a blob.

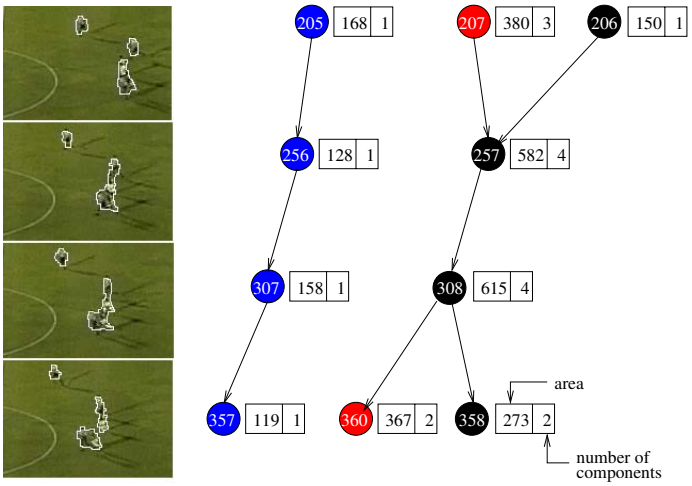


Fig. 4.4: Example of graph construction.

- x, y: the coordinates of the center of a blob in the image.
- *color*: the color associated to a blob (Sec. 4.3.3).

After the graph construction, the following parameters can be defined:

- *num_comp*: the number of components relative to the number of players in a blob (Sec. 4.3.2).
- *dist*: the distance between two linked nodes.
- *direction*: the direction of the players trajectory.
- *velocity*: the velocity of the objects.

The main function of the edges information is to define the path or the possible paths of a player on the graph, during the tracking process. Thus, edges linking very distant blobs, according to the d_{max} value, are not included in the graph. This value may be defined from the maximal displacement a component is supposed to achieve between two consecutive frames.

4.3.2 Number of components definition

The correct determination of the number of components in a blob is important if we need to split this blob (representing more than one player), and make correct decisions about trajectories during the tracking. This number is difficult to determine when, for example, a blob containing more than one player, in frame t, is split or connected to another blob in frame $t + 1$.

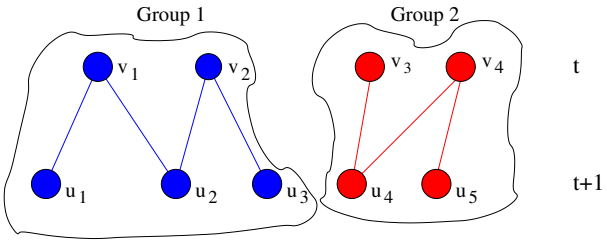


Fig. 4.5: Nodes grouped by common edges.

To solve this problem, we group blobs by considering the edges between them, as shown in Fig. 4.5. Two nodes v_1 and v_2, in frame t, belong to the same group if there is any node u, in frame $t+1$, so that there exist edges (v_1, u) and (v_2, u) in the graph. A simple way to define a group is to use a depth-first search of a new undirected graph defined by nodes and edges in frame t and $t+1$. For each new searching on the graph, starting from nodes in frame t, a new group number is defined. Each visited node, during this searching, receives the corresponding defined group number. For example, in Fig. 4.5, the search starting from nodes v_1 and v_3 defines the groups number 1 and 2, respectively.

The area of the blobs considered here is a parameter associated with the number of components of each node. This area (in terms of number of pixels) is inversely proportional to the distance of the players with respect to the camera location.

For each group of nodes in frame t, belonging to the same group, we consider a subdivision process of the objects in frame $t+1$. First, the number of objects belonging to the same group is defined as the sum of the components number of its nodes. For each node u in frame $t+1$, belonging to a certain group, the area of the corresponding blobs in the actual position (based on the location of the players on the field) is estimated and subtracted from the actual area indicated in the blob. At this point, the total number of components of the group in frame t is decreased by one, and the number of components of the nodes in frame $t+1$ is increased in the same proportion. This step guarantees that each node is associated at least with one object of the scene. The above procedure is executed while the number of components in the region is greater than zero. At each step, we always start from the node having the highest area. The following algorithm summarizes the main steps of this procedure.

Algorithm

1. Determine the number of objects η belonging to group R by adding the number of components, η_v, of each node at frame t.

   $\eta_R \leftarrow \sum_{v \in R}(\eta_v)$

2. For each node u belonging to group R in frame $t+1$:

Estimate the area A_p of the player proportional to its blob position in the image.

$A_u \leftarrow A_u - A_p$ {update the current area of node u}
$\eta_R \leftarrow \eta_R - 1$
$\eta_u = 1$ {initialize the number of components of node u}

3. While $\eta_R > 0$ do
   Find a node u with the highest area A_u
   $A_u \leftarrow A_u - A_p$
   $\eta_u \leftarrow \eta_u + 1$
   $\eta_R \leftarrow \eta_R - 1$

Fig. 4.6 illustrates these steps for the number of components definition.

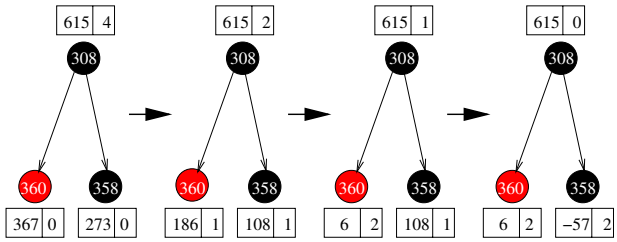


Fig. 4.6: Example of number of components definition.

4.3.3 Blob color definition

In soccer, as in many other group sports, the color of the uniform of the two teams must be different from each other. This information can be used to discriminate the teams as well as to solve occlusion problems. In such a case, some applications consider the histogram back-projection method [73, 47] which can work well for full color uniforms. With our camera positioning setup, it is difficult to discriminate colors and, thus, we work only with the intensity or gray level information of the components.

Also, each part of the uniform may have different colors, as we can see in Fig. 4.8a. Generally, a player can be modeled as a group of many regions, each one having some predominant colors (Fig. 4.7). In [73], for example, the vertical RGB distribution of the blobs (Fig. 4.8b) is compared with the distribution of the corresponding players model.

In our work, since the size of a blob may change, depending on the position of the players on the pitch, we try to divide the model of the players into two or more regions, so that each region represents a part of the team's uniform, i.e., t-shirt, short, socks and

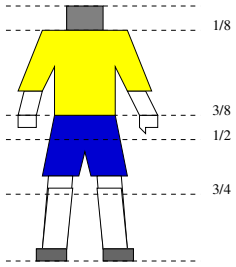


Fig. 4.7: A player model.

so on. For each region, we consider a filtering by threshold based on the vertical intensity distribution of the blobs. The two vertical lines in Fig. 4.8b represent two threshold values, T_1 and T_2, defined as the minimal and maximal mean values of the vertical distribution. These thresholds constitute the limits expressing the significant intensity values of each interest region. Let, for example, R_1 and R_2 be two of these regions related to the t-shirt and short of a player, respectively. These regions can be spatially defined, based on the size of the players in the image and on the model shown in Fig. 4.7, as follows:

$$\frac{1}{8} * psize < R_1 < \frac{3}{8} * psize$$

$$\frac{4}{8} * psize < R_2 < \frac{3}{4} * psize,$$

where $psize$ is the estimated player size according to its position on the pitch.

For each region R_i, we count the number of pixels, p, of a blob lower than T_1 and greater than T_2, yielding the following values

$$S1_{R_i} = \#\{p\} \quad \forall p \in R_i \wedge R_i(p) < T1$$

$$S2_{R_i} = \#\{p\} \quad \forall p \in R_i \wedge R_i(p) > T2$$

These values are associated with the most discriminant intensity values of each region, based on the considered vertical intensity distribution of the blobs. Thus, the color or intensity of each region, R_i, can be defined as

$$C_{R_i} = \begin{cases} 1, & \text{if } \frac{S1_{R_i}}{S1_{R_i}+S2_{R_i}} > 0.6 \\ 2, & \text{if } \frac{S1_{R_i}}{S1_{R_i}+S2_{R_i}} < 0.4 \\ 0, & \text{otherwise} \end{cases}$$

Finally, the color of a blob, representing the team's identification, can be determined from the color information of a region or from a combination of colors of different regions.

For example, for the video sequence considered here, the blobs color is defined as:

$$C_{blob} = \begin{cases} 1, & \text{if } C_{R_1} = 1 \text{ or } C_{R_2} = 2 \\ 2, & \text{if } C_{R_1} = 2 \text{ or } C_{R_2} = 1 \\ 0, & \text{otherwise} \end{cases}$$

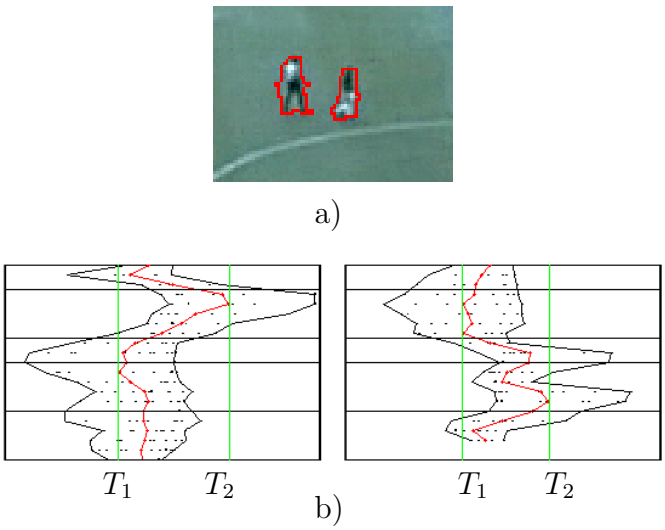


Fig. 4.8: Vertical intensity distribution. a) Two players, and b) their corresponding intensity distribution.

4.3.4 Splitting the blobs

As we have seen in Sec. 4.3.2, we define the number of objects in each blob by considering spatial and temporal information of the segmented objects. In this subsection, we consider the problem of splitting the blobs containing more than one tracked object with the aim of having just one blob associated with one player. Naturally, this splitting can be difficult to implement, mainly in cases when two or more players are very close to each other and the occlusion is almost total.

During the segmentation step, some objects can be grouped into one blob due to small distances between them or effects such as shadows and noise. Our first approach to the splitting problems tries to isolate players linked by short connections, as shown in Fig. 4.12a. This procedure may be seen as a postprocessing of the segmentation step. Further, a blob can be split by considering the constructed graph and the model of the blobs, as we will discuss later.

To illustrate this splitting process, we select regions of a video sequence (Fig. 4.9) containing some simple occlusions. Fig. 4.10 shows the blobs obtained after the segmen-

tation step. Fig. 4.11 shows the corresponding graph and the number of components defined for each node.

a) b) c) d) e) f)

Fig. 4.9: Some cases of occlusions.

a) b) c) d) e) f)

Fig. 4.10: The blobs related to the sequence in Fig. 4.9.

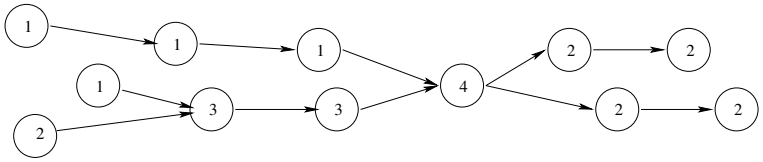


Fig. 4.11: The graph representation of the blobs in Fig. 4.10.

Splitting using morphological operators

To eliminate short connections between blobs, we consider these blobs as regions of the original images and apply a set of morphological operations [63]. First, the original image (Fig. 4.12a) is eroded n times (n depends on the players position on the field). The result of this operation is shown in Fig. 4.12b (the shaded regions). The next step consists of a conditional homotopic thickening of these shaded regions, limited to the corresponding blob contours (Fig. 4.12c).

Fig. 4.15 shows some examples of the splitting by morphological operations of the components in Fig. 4.10.

Splitting using the blobs model.

In this subsection, we consider the model of the blobs and the information conveyed by the graph to split a blob, containing more than one player, which cannot be split by simple morphological operations.

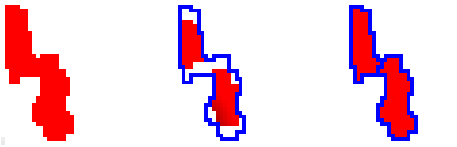


Fig. 4.12: Simple splitting by morphological operations. a) original connected blobs, b) erosion operation, and c) conditional homotopic thickening.

A blob in the current frame can be split into two or more blobs, depending on the number of components of the corresponding node in the graph and on its configuration in the previous frame. Hence, the model extracted from the blobs before an occlusion is considered in the connected version of the current blob, during a matchig operation which further yields a splitting of this blob. For example, in Fig. 4.13, the blobs in the current frame (Fig. 4.13c) were obtained from the shaded blob in Fig. 4.13b and from the model of the isolated blobs in the previous frame (Fig. 4.13a).

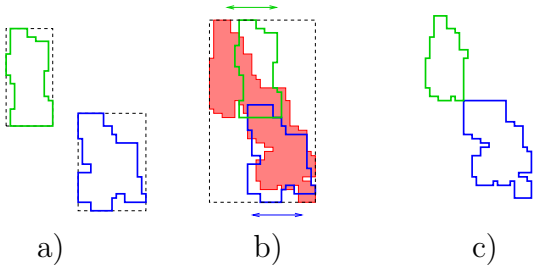


Fig. 4.13: Vertically blob splitting using the blobs model. a) blobs before connection, b) connected blobs, and c) blobs after a vertical splitting.

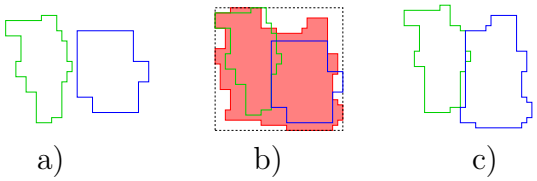


Fig. 4.14: Horizontally blob splitting using the blobs model. a) blobs before connection, b) connected blobs, and c) blobs after a horizontal splitting.

Depending on their location in the previous frames and on the number of components, the blobs can be split vertically or horizontally. Fig. 4.13 and Fig. 4.14 show an example of vertically and horizontally blob splitting, respectively. Since this splitting into more than

three blobs becomes more complex, we consider only two or three components splitting at the same time. There are situations in which the size of a blob is not big enough to be split, horizontally or vertically. In such a case, this blob is not split and we consider that some of its corresponding objects are in total occlusion and, therefore, their location in the image is the same.

As stated before, the graph representation considered here provides useful information for the correct splitting of the blobs. This graph can be constructed in a direct order in which the normal sequence of the events is taken into account or in a inverse order, where we start creating the graph from the last frame. These two ways of constructing the graph allow us to consider the splitting situations before and after a total occlusion.

Fig. 4.16 shows the final configuration of the blobs obtained after a splitting of the the segmented regions in Fig. 4.15.

a) b) c) d) e) f)

Fig. 4.15: Splitting by segmentation.

a) b) c) d) e) f)

Fig. 4.16: Splitting by using the blobs model.

4.3.5 Tracking each player

Calibration of the cameras

In order to get the actual position of the players on the field, we need to calibrate the cameras, i.e, to find their intrinsic (e.g., focal length, lens distortion) and extrinsic (e.g., translation, rotation) parameters allowing the object-image transformation of the players coordinates. In this work, to estimate these parameters we apply the perspective projection matrix which relates the images coordinate to the world coordinate system [6].

The image coordinate system $o - xy$ and the world coordinate system $O - XYZ$ can be expressed using homogeneous coordinates as follows.

$$\begin{bmatrix} xh \\ yh \\ h \end{bmatrix} = C \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.1}$$

C is a 3x4 matrix that performs rotation and perspective projection transformations and h is a scale factor.

After expanding Eqs. 4.1, we get the following two equations:

$$\begin{aligned} &XC_{11} + YC_{12} + ZC_{13} + C_{14} - XxC_{31} \\ &-YxC_{32} - ZxC_{33} - xC_{34} = 0 \end{aligned} \tag{4.2}$$

$$\begin{aligned} &XC_{21} + YC_{22} + ZC_{23} + C_{24} - XyC_{31} \\ &-YyC_{32} - ZyC_{33} - yC_{34} = 0 \end{aligned} \tag{4.3}$$

For the 2D case, with the z coordinate in the world coordinate system equal to zero (assuming that the soccer field is flat), and $C_{34} = 1$ in the homogeneous coordinate system, we have the following equation

$$\mathbf{Ax} = \mathbf{b} \tag{4.4}$$

$$A = \begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -y_1X_1 & -y_1Y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -y_2X_2 & -y_2Y_2 \\ . & . & & & & & & . \\ . & . & & & & & & . \\ . & . & & & & & & . \\ X_n & Y_n & 1 & 0 & 0 & 0 & -x_nX_n & -x_nY_n \\ 0 & 0 & 0 & X_n & Y_n & 1 & -y_nX_n & -y_nY_n \end{bmatrix}$$

$$x^T = [\ C_{11}\ C_{12}\ C_{14}\ C_{21}\ C_{22}\ C_{24}\ C_{31}\ C_{32}\ ]$$

$$b^T = [x_1\ y_1\ x_2\ y_2\ \ldots\ x_n\ y_n\ ]\,,$$

where $(X_i, Y_i, 0)$ represents the world coordinates of the reference point i and (x_i, y_i) is the image coordinate of the same point. By solving Eq. 4.4, we find the 8 necessary parameters for calibration. As each reference point generates two equations, we need at least four of such points. In case of more than four points, the system in Eq. 4.4 can be solved by considering the least-square method. To obtain these reference points, we measure the length of lines in the playing field and detect their corresponding intersection points.

Determination of the players location in the image

The players location in the image is determined by the position of the players feet in the blob. This can be done by finding the maximum y coordinate of the blobs and its middle point in the x coordinate. However, this method fails when lower parts of the player are lost during the segmentation or when shadows are present in the segmented blob, as shown in Fig. 4.17. In this work, we consider this situation by computing the players size based on their current location and their maximal (near the camera) and minimal (far from the camera) estimated size in the image. The y coordinate of a player location is obtained by adding the computed size of this player to the coordinate of the uppermost pixel of the corresponding blob. Here, we consider the fact that by taking into account this blob pixel, we avoid including the shadow of the player in the definition of its location, as illustrated in Fig. 4.17.

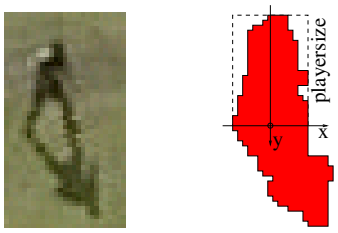


Fig. 4.17: Definition of the player location based on its estimated size in the image.

Since the maximal and minimal size of the player is defined by considering its feet location in the image, the estimated vertical location of the player which matches the expected player size, in the corresponding blob, can be computed as follows. Let y be the vertical position of the feet initially given by

$$y = y_0 + psize_{min},$$

where y_0 is the top vertical pixel of the player blob and $psize_{min}$ is the minimal estimated player size. At each step, the value y is increased until $psize_y \leq y - y_0$, $psize_y$ being the player expected size at the y position defined as:

$$psize_y = psize_{max} - (psize_{max} - psize_{min}) * Y/field_y,$$

where $psize_{max}$ is the maximum estimated player size, $field_y$ is the vertical dimension of the playing field, and Y is the real coordinate of y.

From the image coordinate (x, y), we calculate the player coordinate (X, Y), in the world coordinate system, by substituting the value of (x, y) in Eqs. 4.1, and assuming always that the Z coordinate of the players is equal to zero.

4.3.6 The minimal path searching

The tracking of a player starts by its blob definition and the corresponding node identification in the constructed graph. The color of this blob is used as the team reference for the rest of the tracking. At each step, we traverse the graph by considering a minimal path using the distance information between the blobs. This method represents an easy way to track isolated players since there is only one edge to be considered at each step. If an edge connecting two nodes does not exist, we assume that the player was lost, probably because of an error during the segmentation step. In such a case, we can extrapolate the location of the player in the image, using a prediction method, and we try to find a nearest blob in the next frames. A common example of non detection of a player occurs when it falls down and its size becomes smaller. The tracking of the players in case of contacts or occlusions by other players is more difficult to consider and needs additional information to be implemented. Although the splitting of the blobs (Sec. 4.3.4) is supposed to eliminate this problem, some situations remain in which one node may have more than one player.

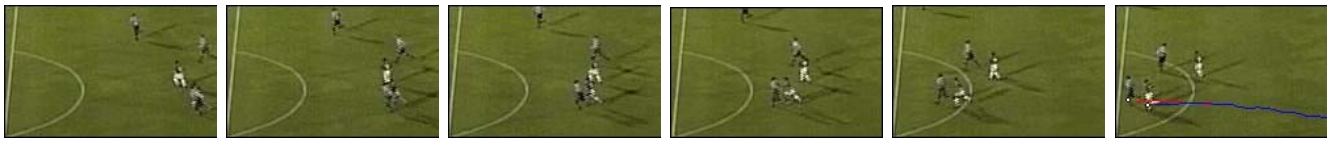


Fig. 4.18: Tracking players. Situation in which two players from different teams are running together.

This work pays special attention to the case of two player occlusions since it is one of the most common situation in soccer (e.g., two players disputing the control of the ball), together with the tracking of the isolated players. Occlusions of more than two players are considered only in case of short temporal contacts.

Soccer, as many other sports, is a tactical game and the success of a team depends also on the way its players tackle the opponent ones. This is why a defender is always very close to the offensive player of the other team. Besides the cases of crossing or passing by, all these events generate occlusions or contacts between two players, Tracking, in such a situation, is performed by taking into account the location of the blob containing these two players and by considering that they share the same trajectory.

To maintain the right trajectory of the players after an occlusion, it is important to correctly identify their nodes in the graph. For this purpose, we consider the color of the blobs (Sec. 4.3.3) as well as the distance information conveyed by the graph. As stated before, one blob color is defined for each team, according to the algorithm described in Sec. 4.3.3. This information is used as a weight w in the determination of a minimal value of the graph edges. Experimentally, we set $w = 1$ when the color of the blob is the

same as the one defined in the beginning of the tracking, and $w = 4$ when these colors are different. If the blob color cannot be identified ($C_{blob} = 0$, as stated in Sec. 4.3.3), we set $w = 2$.

Fig. 4.18 shows examples of two players from different teams running together (the lines indicate the corresponding paths in the figure). In such a case, it is difficult to correctly split the blobs and assign only one trajectory for both players. After a certain number of frames, these connected components are separated and have their own trajectory. This trajectory definition is possible by considering the color of the blobs associated with the corresponding players.

Finally, we need to take into account situations in which this blob color information does not allow a correct tracking, as it happens, for example, when the players in contact belong to the same team (Fig. 4.19). If this contact is short, in terms of number of frames, we maintain the same direction for each player as before the occlusion. Further, if the contact is long and without much motion then, heuristically, we can consider that each player moves in opposite directions after an occlusion event.

Fig. 4.19: Tracking players. A complex situation representing occlusions of players from the same team.

At each step of the tracking method, the real 2D coordinates of the players on the field is reconstructed, using the calibration parameters and the image coordinates representing their location in the image. Initially, it is also defined the field of view of each camera. Based on this information, we can determine exactly the cameras a tracked player is visible from and choose the one that better focus on this player. The overlapped region, i.e, the region visible by more than one camera, is also considered here in the solution of occlusion problems. For example, if a tracked player in this overlapped region appears occluded in the current camera, then we can verify if this player becomes isolated in another view. This kind of solution is also explored in [36] which considers 8 cameras covering the whole goal area.

4.4 Applications and some results

To illustrate the proposed method, we consider a game of the Brazilian soccer league filmed by four digital cameras placed at one side of the pitch and at the highest location of the stadium. Position and distance between cameras were chosen arbitrarily so that

one camera covered at least one fourth of the playing field and had extra overlapping regions. Fig. 4.20 shows these cameras placement.

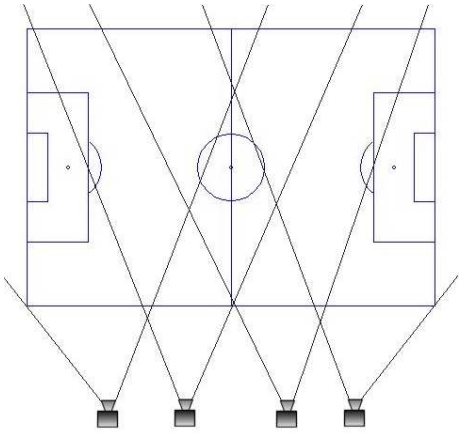


Fig. 4.20: Example of cameras placement.

By considering four views, we ensure that we have enough information to calibrate the cameras, while preserving good features (e.g., size and color) of the players. Here, for example, we can easily remove noise from the processed images based on the size of the segmented players. Also, as we have seen before, enough color information from the segmented objects can be very useful in the tracking process, mainly when two or more players are in contact.

The above mentioned game was filmed in the afternoon of a sunny day with a partially clouded sky. The results discussed here concern a 10 minute game processed by the segmentation algorithm described in Sec. 4.2. Further, the tracking method was performed for the 22 players from both teams. Table 4.1 illustrates some numerical results of the test. The second column shows, for each player in column 1, the number of occlusions or contacts correctly processed by the method, i.e, the cases in which the blobs were correctly tracked during the occlusion events. The third column shows the situation in which the tracking failed because of two or three player occlusions. The fourth and fifth columns show the case of fails caused by occlusions of more than 3 players and by problems such as segmentation errors or by players out of the playing field. The sixth column shows the total number of stops during the tracking caused by all the above mentioned events.

Finally, the seventh column shows the total number of frames whose components tracking was done manually due to errors in the automatic tracking. This manual process consists in following the players position on the screen by a displacement of the mouse cursor whose position on the successive frames is automatically detected. Normally, in case of two or three player occlusions, the number of manual corrections is low (only 10% of the total number of manual corrections). In case of occlusions of more than three

Player	Number of solved occlusions or contacts	Number of non-solved occlusions 2-3players	Number of non-solved occlusions 3+ players	Number of lost players	Number of stops during the tracking	Number of frames manually tracked	Number of frames with correctly solved occlusions
1	8	1	0	0	1	1	107
2	38	8	4	0	12	418	446
3	38	3	3	0	6	254	640
4	51	7	4	0	11	111	664
5	52	8	2	0	10	318	672
6	52	8	2	4	14	104	721
7	36	2	1	2	5	48	422
8	47	7	5	1	13	399	516
9	55	9	2	1	12	345	1229
10	57	5	4	2	11	353	585
11	49	5	6	0	11	492	1196
12	2	1	2	0	3	45	27
13	47	2	4	4	10	368	1028
14	44	3	4	0	7	428	822
15	26	4	4	0	8	267	655
16	51	12	4	0	16	469	538
17	23	4	3	1	8	79	315
18	52	3	3	4	10	187	718
19	52	6	2	1	9	67	544
20	60	4	2	4	10	391	796
21	43	4	3	2	9	179	634
22	59	9	1	0	10	111	830
mean	46.6	5.7	3.1	1.3	10.1	270	699

Table 4.1: Evaluation of the tracking algorithm for a 10 minute sequence.

players caused, for example, by corners or free kicks, the number of manual corrections is higher (more than 90% of these corrections). During the ten minutes of the analyzed sequence, there were two corners and three free kicks (Fig. 4.21). The final column in Table 4.1 shows the number of frames having occlusion events correctly solved. The last line of this table shows the mean value for each considered case, by taking into account all the players, except the goalkeepers (players number 1 and 12). These players were excluded from the mean because of the few number of occlusions concerning them (as we can see in the second, third and fourth columns in Table 4.1), and the success of the tracking method, in such a case (99%), with only two stops on average.

If we consider that the number of solved occlusions is represented mostly by the cases of two or three players in contact, then we correctly solved 89% of these cases for the analyzed video sequence. The mean number of stops (last line in Table 1) shows that during the tracking of each player we stopped one time each one minute, i.e, each 450 frames. By considering the total number of stops and the number of solved occlusions, we have a reduction of 82% of the number of stops in the tracking process, as we can see in Table 4.2. The mean number of manually tracked frames represents only 6% of the total number of frames (Table 4.3), and from 94% of the automatically tracked frames, 15% correspond to cases of occlusions.

	Number of solved occlusions or contacts	Number of stops during the tracking
mean	46,6	10,1
percentage	82%	18%

Table 4.2: Number of stops reduced by this method.

The quality of the images is very important when we deal with occlusion problems, the size of the represented players being proportional to their distance from the camera. For example, in Table 4.1, we can see that we stopped 14 and 5 times for the players 6 and 7, respectively. By analyzing these players location on the field, we remark that they are wing players: one on the left side (the upper region of the images) and the other on the right side of the field (the lower region of the images), defining smaller and bigger representations of these players, respectively. Hence, a straightforward way to improve the whole tracking method proposed here is by the use of multiple cameras positioned on both sides of the playing field.

	Number of manually tracked frames	Number of automatically tracked frames (cases of occusions)	Number of automatically tracked frames (cases of isolated players)
mean	270	699	3531
percentage	6%	15,5%	78,5%

Table 4.3: Number of automatically tracked frames.

Fig. 4.21: Examples of corner and free kick in the video. In such situations, some players were manually tracked.

4.5 Conclusions

In this work, we proposed a method for tracking soccer players which considers many static cameras. Our main aim consists in finding the 2D location of the players on the field at each instance of time. The tracking algorithm is based on the search of paths in a graph defined by blobs representing segmented players. The different cases of occlusions or contact of these players are treated by splitting the corresponding blobs and by taking into account information such as color, area of the blobs, players trajectory and so forth. This algorithm has been used in a large number of soccer video images and its performance was illustrated here through a 10 minute processing whose basic statistical results show the effectiveness of the method.

Capítulo 5

Desenvolvimento de um ambiente visual para o processamento de seqüência de imagens

Neste capítulo apresentamos os detalhes sobre a funcionalidade, desenvolvimento e implementação de um ambiente visual flexível, que possibilita a montagem de algoritmos de processamento de seqüência de imagens e a adição de novas funções de uma forma relativamente fácil.

Um ambiente de programação visual permite a construção de programas ou algoritmos através da seleção de ícones ou figuras e conexões que definem o controle e fluxo de dados entre eles. Este tipo de ambiente tem sido bastante empregado no desenvolvimento de protótipos rápidos de programas, assim como ferramenta de auxilio para as pessoas com pouco conhecimento de programação. Outra vantagem é a possibilidade de execução paralela de funções independentes e a manutenção de cada função como um módulo independente.

O ambiente desenvolvido neste trabalho considera os principais aspectos de uma linguagem de programação visual e utiliza o modelo de fluxo de dados para determinar a ordem de execução da seqüência de operações. As funções da interface são representadas por figuras em forma de caixas retangulares e as ligações entre as caixas representam o fluxo de dados.

Uma das dificuldades de um ambiente de programação visual é a implementação do controle que permita iterações no algoritmo. Neste ambiente, foram desenvolvidos os mecanismos que controlam as iterações, tais como laços e realimentação, usando a teoria de grafos, na definição de dependência funcional e na detecção dos laços externos e internos.

Neste capítulo apresentamos ainda exemplos de construção de algoritmos usando as ferramentas desenvolvidas. Um dos exemplos ilustra a construção do método de rastrea-

mento de marcadores proposto no Capítulo 2. Este ambiente foi usado extensivamente na segmentação dos jogadores de futebol discutida no Capítulo 4.

Desenvolvimento de um ambiente visual para o processamento de seqüência de imagens

P.J. Figueroa N. J. Leite

Instituto de Computação
Universidade Estadual de Campinas
{pascual,neucimar}@ic.unicamp.br

R. M. Barros

Faculdade de Educação Física
Universidade Estadual de Campinas
ricardo@fef.unicamp.br

5.1 Introdução

Nos últimos anos tem sido crescente o número de sistemas ou ambientes que permitem a programação visual. O termo de programação visual refere-se a possibilidade de construir programas ou algoritmos através da seleção de ícones ou figuras que representam as funções, e as conexões entre elas definem o controle e o fluxo de dados. O interesse pelo desenvolvimentos de tais sistemas deve-se ao fato de que a programação visual é mais intuitiva do que a textual podendo, assim, ser usada por pessoas com pouco conhecimento de programação.

O termo de programação visual tem sido empregado igualmente em ambientes de linguages de programação como, por exemplo VisualWorks do ParcPlace, PowerBuilder do Powersoft, Visual Basic e Visual C++ da Microsoft, etc. Nestes ambientes, os componentes gráficos são usados apenas na construção da interface do programa, enquanto que para se gerar o programa propriamente dito é usada uma linguagem de programação textual. Já os sistemas com linguagem de programação visual, tais como LabVIEW [71], Cantata [56], VEE [34], Show and Tell [40], Prograph [14], VIPER [51], etc., permitem a construção de programas apenas com figuras ou ícones e textos adicionais.

Os ambientes de programação com linguagem visual podem ser diferenciados ainda pelo nível de programação ou campo de aplicação. Os ambientes que permitem programação de baixo nível são geralmente de propósito geral, ou seja, podem ser usados para diferentes objetivos e com possibilidades de gerar desde programas simples até programas mais complexos. Exemplos deste ambientes são Show and Tell [40] e Prograph [14]. Por outro lado, os ambientes que têm uma aplicação ou uma area de domínio específica, tais como o Cantata que é destinado essencialmente ao processamento de imagens, o Labview, apropriado à construção de laboratórios virtuais de instrumentação etc., geram programas de alto nível.

Um dos problemas das linguagens de programação visual de baixo nível e em geral da programação visual refere-se ao tamanho que a descrição de um algoritmo pode vir a

ocupar na área de visualização. Para amenizar este problema alguns ambientes oferecem a possibilidade de usar a noção de procedimentos agrupando várias funções num só bloco ou figura.

Um dos paradigmas mais usados no desenvolvimento de ambientes de programação visual é o modelo de fluxo de dados [62], o qual consiste em construir um grafo direcionado onde os nós representam as funções e os arcos, o fluxo de dados entre as funções. O controle do fluxo de dados é realizado seguindo o principio do data-flow, ou seja, cada nó é executado assim que as entradas correspondentes estiverem disponíveis. Um outro modelo de execuçao é o data-driven que consiste da execução da função associada a um nó somente quando esta for requisitada.

Algumas das principais vantagens das linguagens de programação visual baseadas em fluxo de dados são as seguintes:

- permitem a execução paralela dos nós independentes.
- podem ser usadas para gerar um protótipo rápido de programas.
- permitem a modularidade, ou seja, podem ser facilmente estendidas com novas funções.

Atualmente existem várias diferenças nas propriedades das linguagens de programação visual baseadas em fluxo de dados. As seguintes características podem ser consideradas [62]: o estilo de representação, suporte de iterações, disponibilidade de funções de alto nível , conceitos de tipos e modos de execução, etc.

O modelo de fluxo de dados puro permite apenas fluxo seqüêncial de dados (grafos acíclicos). Assim, para que uma linguagem visual possa lidar com laços ou iterações (grafos cíclicos), é necessário incluir algum controle ao fluxo de dados. Alguns ambientes simplesmente não consideram estes casos [41], outros tratam este problema incluindo certos artifícios ou controles adicionais [34, 51] ou a partir de uma recursão [14, 40, 56, 71], o que torna a concepção de laços menos natural. Estas iterações ou laços são fundamentais numa linguagem de programação no caso, por exemplo, em que se deseja repetir uma mesma seqüência de operaçõs sobre diferentes dados de entrada ou fazer uma atualização baseada em um cálculo de dados anteriores. Por este motivo, é importante que uma linguagem visual inclua mecanismos de controle do fluxo de dados numa estrutura iterativa do tipo laço.

Neste trabalho, foi desenvolvido um ambiente visual para a construção de algoritmos para o processamento de seqüência de imagens denominado VISPE (A Visual Image Sequence Processing Environment). O ambiente desenvolvido considera os principais aspectos de uma linguagem de programação visual e utiliza o modelo de fluxo de dados para determinar a ordem da seqüência das operações. As funções que constituem um algoritmo

são representadas através de uma figura em forma de uma caixa retangular, com o nome da função sobre as mesmas, e as ligações entre as caixas representando o fluxo de dados através das funções. Cada função é um módulo independente que, baseado nos tipos de entrada, saída e parâmetros, executa uma determinada tarefa. Este módulo é equivalente a uma função *overloading* na linguagem orientada a objetos. Desta forma, várias funções que executam as mesmas tarefas, mas com diferentes tipos de dados, podem ser representadas numa única caixa. Outro aspecto importante do ambiente desenvolvido refere-se à disponibilidade de controle de laços indispensável, por exemplo, na construção de algoritmos sobre seqüência de imagens. Como ilustrado posteriormente, tais algoritmos necessitam da definição de laços internos, assim como do conceito de realimentação (*feedback*).

O restante deste trabalho está organizado da seguinte forma: na próxima seção será detalhado o procedimento para determinar a seqüência de execução das funções, assim como o controle, em casos de iterações. A seção 3 aborda a definição dos tipos e fluxo de dados e a seção 4 descreve a interface do ambiente. A seção 5 discute alguns exemplos de aplicação e, finalmente, a seção 6 apresenta algumas conclusões.

5.2 Controle de fluxo de dados e ordem de execução

Para se determinar a ordem de execução da seqüência de funções é considerado uma ordenação topológica a partir do grafo, através de uma busca em profundidade [67]. Este ordenamento topológico define a dependência dos dados, o que garante que as entradas de dados de uma função estarão disponíveis e atualizadas antes da sua execução. O grafo é construído considerando os nós ou vertices como as funções ou caixas e os arcos ou arestas representam as conexões entre as caixas ou funções.

No inicio da execução do algoritmo, as funções sem dependência são inseridas numa fila de processamento. A seguir, cada uma destas funções são extraídas da cabeça da fila para ser executada. Se uma função for executada com sucesso, todas as outras funções que são dependentes desta são inseridas na fila, de tal forma a mantê-la sempre ordenada pela dependência funcional. Os resultados gerados ficam disponíveis para serem lidos por estas funções dependentes. A figura 5.1 mostra um exemplo de uma seqüência de operações representadas por caixas, e a conexão entre elas representa o fluxo dos dados. O grafo construído (figura 5.2), a partir desta figura, é acíclico e, portanto, a ordem de execução das operações pode ser facilmente definida considerando-se uma ordenação topológica.

Como mencionado anteriormente, para se trabalhar com uma seqüência de imagens é importante contar com ferramentas de controle de laços e iterações. Neste ambiente, estes laços são controlados diretamente utilizando apenas duas caixas: *LoopFor* e *CondIf*, sem nenhum controle adicional, como ilustra a figura 5.3.

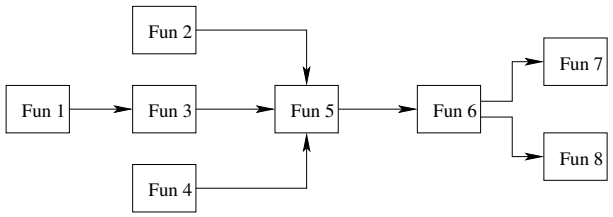


Fig. 5.1: Exemplo de uma seqüência de operações, em que cada caixa representa uma função.

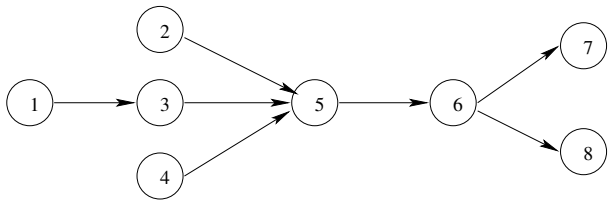


Fig. 5.2: Grafo construído apartir da dependência funcional entre as caixas da figura 5.1 .

A caixa *LoopFor* representa um contador contendo duas portas de saída e uma de entrada. Pela primeira porta é transmitido o fluxo dos dados e pela segunda, um valor booleano que determina se o loop chegou ao final. Esta caixa possui três parâmetros definidos antes da execução. O primeiro é o valor inicial do contador, o segundo, o valor final e o terceiro, o valor do incremento ou decremento. A caixa *CondIf* pode ser vista como uma chave que deixa passar o fluxo de dados apenas se uma segunda entrada boleana o permitir, i.e, se o seu valor booleano for verdadeiro. Durante a execução, o contador da caixa *LoopFor* é incrementado ou decrementado a cada iteração. Ao final do loop, nenhuma saída é gerada na primeira porta e, conseqüêntemente, nenhuma função conectada a esta saída é inserida na lista. Neste caso, a saída da segunda porta gera o valor booleano verdadeiro, o que ativa a caixa *Condif* a passar os dados adiante.

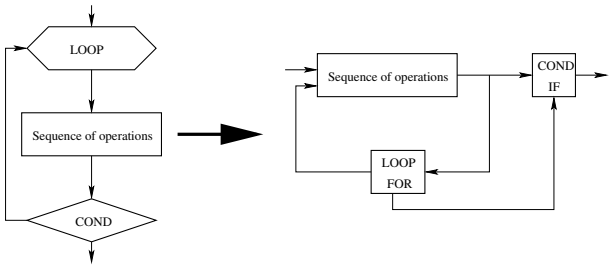


Fig. 5.3: Construção de laços no ambiente desenvolvido.

A definição da dependência funcional em presença de laços é mais dificil do que numa

seqüência simples. O problema principal no controle de laços é o de como "quebrar" o laço sem quebrar esta dependência funcional. Para resolver este problema, neste ambiente, os laços são quebrados, não incluindo a conexão na entrada da função *LoopFor*, durante a construção do grafo, como ilustra a figura 5.4. Mas como todas as caixas são executadas só se suas entradas estiveram prontas, uma exceção precisa ser considerada aqui quando do início da execução do algoritmo.

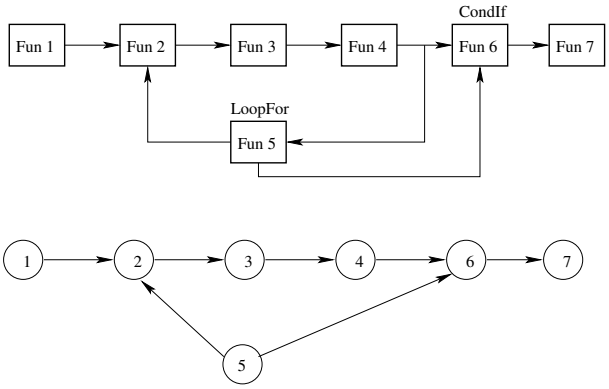


Fig. 5.4: Construção de dependência funcional em caso de laços.

O controle de fluxo em iterações com laços aninhados (figura 5.5) é mais complexo, principalmente pela necessidade de se inicializar os laços internos a cada iteração do laço externo. Este problema pode ser resolvido incluindo-se uma opção de "auto-reset" para os laços internos, cada vez que seu contador chegar ao final. No entanto, ainda existe o problema de como os mesmos podem ser inseridos na fila, na próxima iteração do laço externo, já que não existe mais nenhuma dependência entre as caixas *LoopFor*, pois a entrada destas é desativada quando da construção da dependência funcional.

A solução adotada para este problema foi determinar o laço externo, para cada laço interno a partir de um novo grafo, cuja construção inclui todas as conexões do esquema. Para cada nó *LoopFor* é realizada uma busca em profundidade encontrando-se todos os caminhos a partir deste nó até ele mesmo. Todos os outros nós *LoopFor* encontrados no caminho são considerados laços internos deste nó. Desta forma, cada vez que um laço externo for executado e o contador ainda não tiver chegado ao final, todas as caixas *LoopFor* internas são inseridas na fila. O algoritmo para determinar os laços internos e externos de uma seqüência de operações pode ser resumido da seguinte forma. As cores mencionadas a seguir visam uma melhor ilustração das diferentes etapas.

Algoritmo 1: FindLoops

1. Construir um grafo a partir das caixas e conexões e fazer uma lista de caixas que representam os laços.

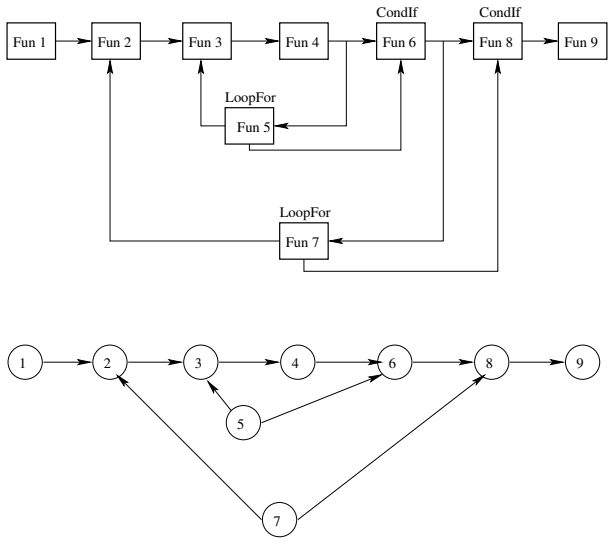


Fig. 5.5: Construção do grafo e dependência funcional em caso de laços aninhados.

2. Para cada laço faça :

    3. etiquetar com cor branca todos os nós
    4. precorrer o grafo, fazendo uma busca em profundidade e a cada passo etiquetar os nós com cores verde e vermelho. Se existir um caminho de volta que passa por um nó, este é etiquetado com verde, caso contrário, com vermelho.
    5. para cada etiqueta verde verificar se existe alguma aresta conetada a uma caixa de laço e adicionar este laço na lista de laços internos ao laço atual.

6. A partir da lista dos laços internos, determinar o laço externo para cada laço interno.

O exemplo de execução deste algoritmo, para o grafo da figura 5.5, é mostrado na figura 5.6.

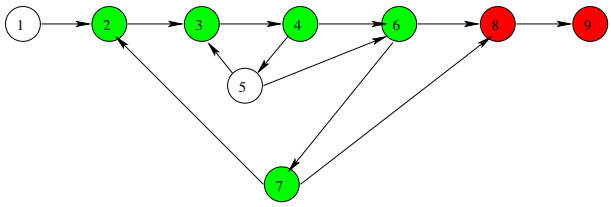


Fig. 5.6: Deteção de laços internos.

Os casos de caixas com realimentação podem ser tratados como se fossem um laço interno com apenas uma iteração, já que as caixas *LoopFor* podem gerar como resultado o valor do contador ou deixar passar os dados de entrada.

A algoritmo completo para o controle de fluxo de dados é descrito a seguir.

Algoritmo 2: Controle de fluxo

1. Testar a compatibilidade dos dados de entrada e saída.
2. "Resetar" entradas e saídas e buffer global de dados.
3. Fazer uma busca para determinar laços internos e externos (**FindLoops**, Algoritmo 1)
4. Inicializar a fila de processos
5. Determinar a ordem dos processos segundo a dependência de dados:
    5.1. construir o grafo sem as arestas conectando os laços.
    5.2. Realizar uma busca em profundidade.
    5.3. Realizar uma ordenação topológica.
    5.4. Incluir os processos independentes na fila dos processos.
6. Enquanto a fila não estiver vazia repetir:
    7. Retirar o primeiro processo da fila.
    8. Testar se as entradas estão disponíveis.
    9. Em caso afirmativo, executar a função correspondente, senão enviar mensagem de erro e concluir o processo.
    10. Em caso de erro na execução, enviar mensagem e concluir.
    11. Se os dados de saída foram gerados:
        12. Passar os dados para as caixas conectadas.
        13. Adicionar na lista de processos as caixas conectadas.
        14. Se o processo atual for um laço, procurar os laços internos deste e adicionar à lista de processos.

5.3 Definição de tipos e fluxo de dados

Como este ambiente foi desenvolvido basicamente para processamento de seqüência de imagens, foram definidos apenas alguns tipos para tratar imagens, números inteiros, matrizes e listas. No entanto, outros tipos podem ser facilmente inseridos quando da introdução de novas funcionalidades no sistema.

Os dados são definidos usando o conceito de herança e overloading da programação orientada a objetos. Este conceito permite um fluxo de dados de diferentes tipos pelo mesmo canal, sendo todos derivados do mesmo pai. O pai é apenas uma clase abstrata cuja única função é fazer a junção de diferentes tipos, como mostra a figura 5.7.

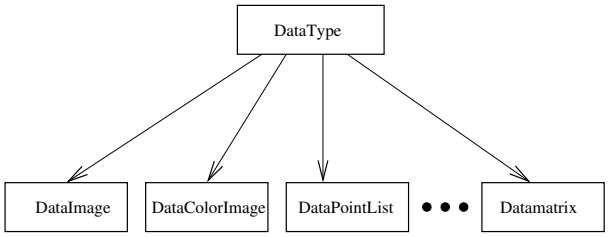


Fig. 5.7: Hierarquia de tipos de dados.

A transferência de dados entre as caixas dá-se através da memória. Quando uma caixa é executada com sucesso, as referências dos dados de saída são copiadas para as entradas das outras caixas conectadas a esta. A partir das referências de entrada, os dados são recuperados da memória, fazendo a transformação para o tipo de dado requerido. Neste caso, não é necessário fazer uma cópia para cada caixa conectada, o que evita uma sobrecarga adicional da memória. Este mecanismo pode causar uma alteração dos dados originais, caso ocorra nas caixas conectadas. Portanto, se alguma transformação é considerada sobre os dados de entrada, esta alteração deve ser realizada sobre uma cópia destes dados.

Os dados também podem ser transferidos através de buffers globais sem a necessidade de uma conexão. Este modo de fluxo de dados é util principalmente para o compartilhamento de dados entre diferentes aplicações.

Sempre que os resultados de uma caixa estiverem disponíveis, estes são copiados para um buffer da memória alocada e ficam disponiveis para serem lidos por qualquer outra caixa. Um número de referência para cada buffer é usado para acessar os dados requeridos, o qual deve ser selecionado com antecedência. Se os dados de entrada do buffer não estiverem disponíveis, durante uma execução, uma messagem de erro será fornecida. Naturalmente, deve haver uma compatibiblidade de tipos na entrada e saída do buffer com a mesma referência.

5.4 Descrição da interface

Nesta interface, as caixas são interconectadas de forma natural como se fossem um circuito digital, podendo-se ainda selecionar o tipo de conexão mais adequado a uma melhor visibilidade. Uma caixa é conectada a outra com apenas dois clicks do mouse: o primeiro

para marcar o ponto inicial e, o segundo, para definir o ponto final do fluxo de dados. Ainda para efeito de visualização, a figura representando estas caixas pode ser movida e rotacionada de 90 graus. Para se conhecer a posição das entradas e saídas (alteradas depois de uma rotação), as figuras possuem uma pequena marca (figura 5.8), indicando a primeira entrada e a primeira saída como ponto de referência. Para verificar se existem dados disponíveis na entrada ou saída das funções existe, ainda, um pequeno ponto de cor vermelha no pino da entrada ou saída.

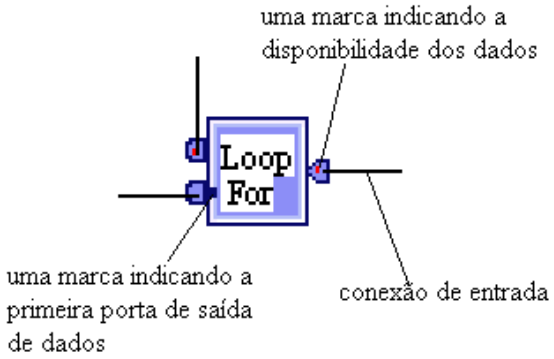


Fig. 5.8: Representação gráfica de uma caixa.

O usuário monta um algoritmo selecionando as funções e conectando as ligações. Antes da execução é necessário definir os tipo de dados de entrada e saída para todas as caixa, assim como inicializar seus respectivos parâmetros. Os tipos de dados e os parâmetros dependem da funcionalidade de cada caixa. A função predefinida *FilterLow*, por exemplo, que executa uma limiarização da imagem de entrada, pode ser aplicada tanto para imagens em níveis de cinza quanto para imagens coloridas nas 3 bandas RGB. Neste caso, as entradas e saídas de cada caixa têm uma lista de tipos para esta função, como mostra a figura 5.9. Da mesma forma, existe um diálogo para definir o valor do parâmetro limiar utilizado na filtragem.

A figura 5.9 mostra, ainda, o uso de um buffer global no compartilhamento dos dados. A caixa selecionada e o número 5 na primeira saída da função indicam que os dados resultantes da execução do *FilterLow* devem ser copiados também para o buffer 5, alem de serem transmitidos para as caixas conectadas a esta.

A seguir são descritos alguns exemplos de funções predefinidas, seus tipos de entrada e sua funcionalidade.

5.4.1 Caixas de entrada e saída de dados

- **ImageSource.** Carrega um arquivo de vídeo ou uma seqüência de imagens.

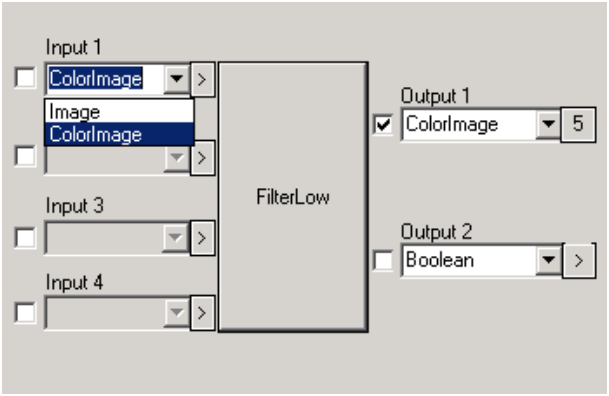


Fig. 5.9: Diálogo para a seleção dos tipos de entrada e saída e de um buffer global.

- **LoadImage.** Carrega um quadro de vídeo.
- **DVCap.** Acessa o driver de captura para carregar as imagens de vídeo e áudio. Esta caixa permite criar aplicações em tempo real.
- **ShowImage.** Mostra a imagem em níveis de cinza ou colorida.
- **SaveData.** Salva dados em diferentes formatos.

5.4.2 Caixas de Controle

- **LoopFor.** Permite construir iterações em forma de laço. Utiliza um contador, uma entrada e duas saídas de dados. A cada iteração, o contador é incrementado ou decrementado em *mStep* (parâmetro fornecido durante a initizialização). Uma das saídas gera dados do tipo booleano, para sinalizar o final da iteração. A outra saída da caixa permite o fluxo de qualquer outro tipo de dado.
- **Counter.** Tem a mesma funcionalidade que a caixa *LoopFor* mas apenas com uma saída e uma entrada. Ao final do número máxino de iteração indicado, o contador gera o valor verdadeiro, ativando o processamento das caixas conectadas a esta caixa de controle.
- **CondIf.** Bloqueia ou deixa passar os dados da primeira entrada, dependendo do valor na segunda entrada que é do tipo booleano. Este controle é geralmente empregado com caixas do tipo *LoopFor* ou *Counter*.
- **Merge.** Deixa passar uma das possíveis entradas, dando prioridade a segunda. Este controle é empregado, geralmente, em casos de realimentação.

5.4.3 Caixas de processamento de imagens

- **Erosion.** Efetua a operação de erosão para imagens em níveis de cinza ou nas bandas RGB para imagens coloridas. Permite selecionar o tamanho da máscara para definir o elemento estruturante, assim como o número de iterações.
- **Gray.** Gera uma imagem em níveis de cinza selecionando uma das bandas RGB de imagens coloridas ou realziando uma média ponderada destas bandas.
- **FilterLow.** Executa uma limiarização da imagem, fazendo com que todos os seus valores menores que um determinado limiar sejam substituídos por zero. Aceita, na entrada, tanto imagens em níveis de cinza como imagens coloridas.

5.4.4 Caixas de transformação de imagens

- **CutImage.** Extrai uma região selecionada da imagem. Como entrada recebe uma lista de pontos e a imagem original.
- **Rotation.** Realiza uma rotação de uma imagem de entrada.
- **SelectRegion** Seleciona uma região de interesse de uma imagem de entrada.

5.4.5 Visão computacional

- **KalmanFilter.** Empregado na predição da posição de objetos em movimento.
- **TempMatch.** Realiza um casamento de padrões para detectar uma determinada região da imagem, semelhante ao modelo de um objeto de interesse.
- **CalibCam.** Executa a calibração de câmeras.
- **Trans2D.** Reconstrói as coordenadas 2D de um ponto na imagem usando a calibração das câmeras.

5.5 Exemplos de aplicação

A figura 5.10 apresenta um exemplo de uma aplicação simples, executada em tempo real, para a deteção de objetos em movimento. A primeira caixa *DVCap* captura uma imagem colorida e envia para a caixa *DetectMotion* onde mudanças de intensidade para cada pixel são detectadas. A imagem resultante é convertida para níveis de cinza em *Gray*, seguida de uma filtragem em *FilterLow*. Finalmente, a caixa *ShowImage* apresenta o resultado. A caixa *LoopFor* está associada ao tamanho da seqüêcia a ser considerada.

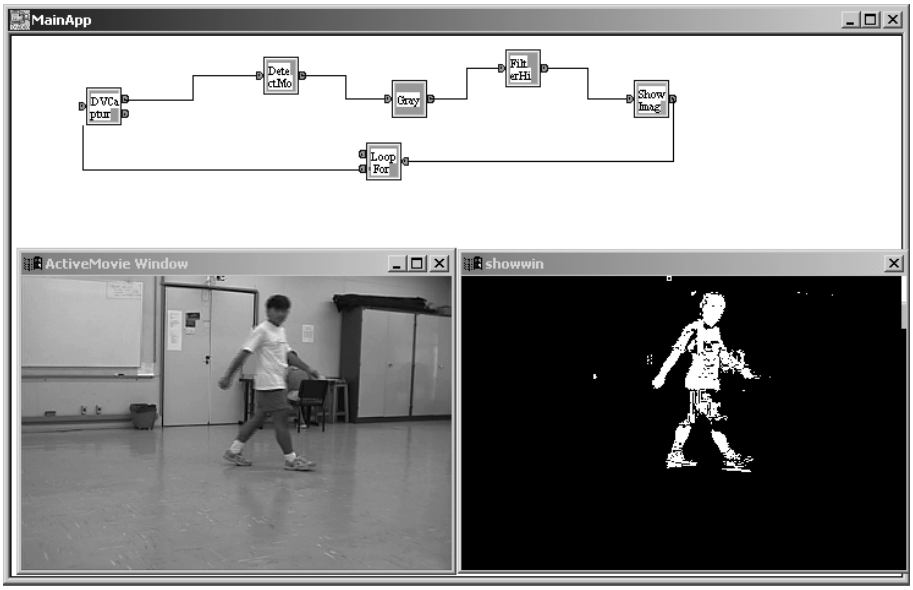


Fig. 5.10: Exemplo de uma aplicação em tempo real usando a interface visual.

A figura 5.11 ilustra outro exemplo de um algoritmo mais complexo construído utilizando as ferramentas do ambiente desenvolvido. Este algoritmo é utilizado no rastreamento de marcadores [25] e consiste da extração de um modelo do marcador, a partir do primeiro quadro (caixas *ImageSource, SelectRegion, CutRegion*), a ser comparado (*TemplMatching*) com outras partes da região de busca (definidas pela caixas *ImageSource* e *SelectRegion*) extraída da imagem dos quadros seguintes (caixas *ImageSource, LoadImage* e *CutRegion*), para se determinar a posição deste marcador nos próximos quadros.

A região de busca é segmentada (*Gray, Inverse, erosion, GetMarker*) para melhorar o rastreamento e a precisão usando ferramentas de morfologia matemática. Para a predição da posição do marcador e definição da região de busca (*SearchRegion*) é utilizada a função de Extrapolação (*ExtrPolation*). A região de busca é atualizada através da caixa *Merge* -caso de realimentação- construído a partir das caixas *LoopFor* e *CondIf*. O Resultado das coordenadas do marcador é armazenado num arquivo (*SaveData*) e visualizado ao mesmo tempo através da caixa *ShowData*.

5.6 Conclusão

Neste trabalho foi desenvolvido um ambiente visual iterativo que possibilita uma descrição natural dos algoritmos de processamento de seqüência de imagens, em particular para segmentação de jogadores. Uma das razões para a implementação deste tipo de interface foi a necessidade de se acoplar os differentes componentes desenvolvidos neste trabalho

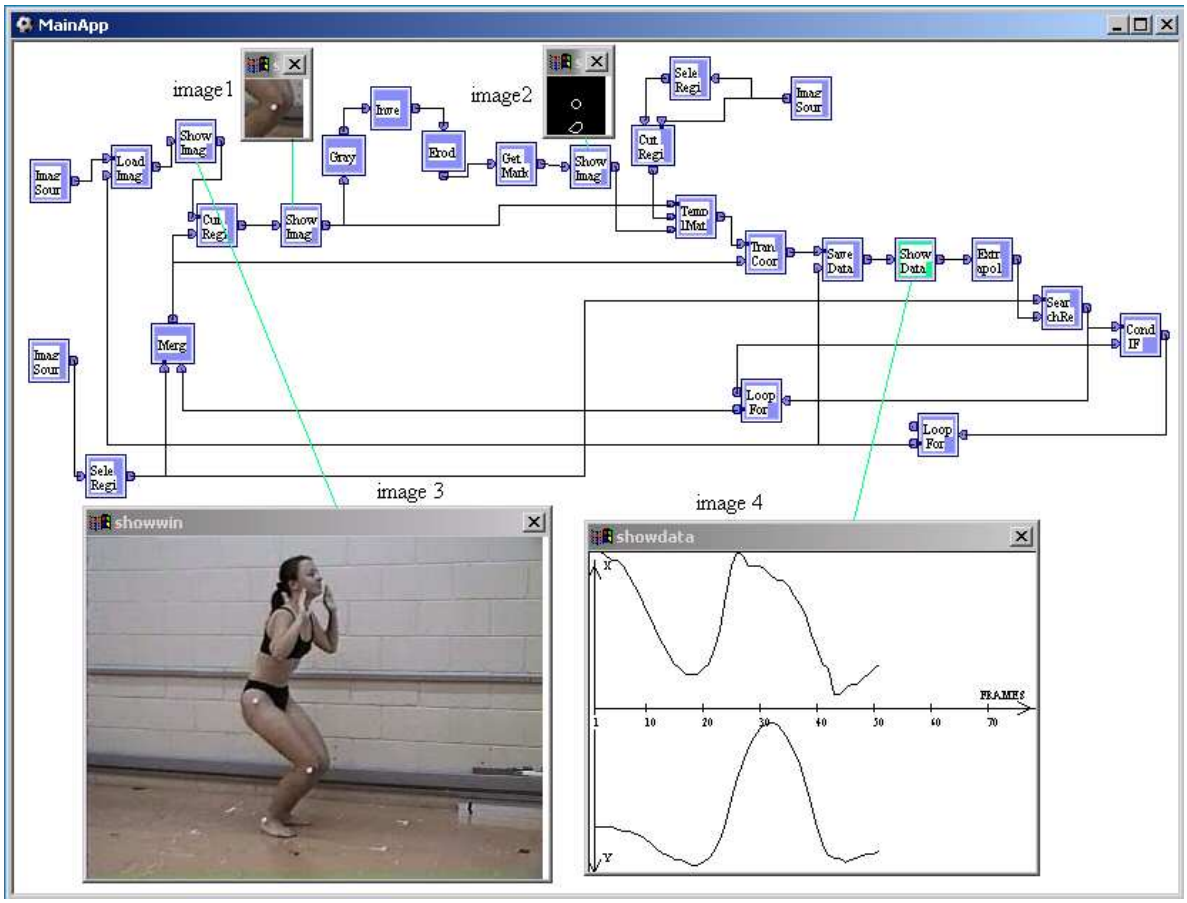


Fig. 5.11: Exemplo de aplicação para tracking de marcadores.

da forma mais simples possível, mantendo cada caixa ou ferramenta como um módulo independente. Esta abordagem possibilita a construção de algoritmos de acordo com as particularidades de cada problema, assim como o aproveitamento de muitas bibliotecas disponíveis no Matlab da Matworks, DirectX da Microsoft, OpenCV da intel, sem que se tenha de mudar a estrutura global do ambiente e o controle do seu fluxo de dados.

Capítulo 6

Conclusão

Neste trabalho foram abordados problemas relativo ao rastreamento de objetos em movimento a partir de seqüências de imagens capturadas por câmeras de vídeo. O objetivo principal do rastreamento foi determinar a posição real dos objetos em movimento visando o cálculo de variáveis cinemáticas, tais como distância, velocidade e aceleração, etc. para uma análise quantitativa deste movimento.

Foram desenvolvidos algoritmos genéricos para o rastreamento de objetos baseado nas características da imagem, tais como intensidade, contraste e forma das regiões representando os objetos de interesse. O método proposto foi implementado em um ambiente de interface e foi aplicado para o rastreamento de marcadores com o objetivo de analisar diferentes tipos de movimentos humanos.

Visando-se considerar o rastreamento de múltiplos objetos, principalmente em situações de oclusão, foi desenvolvido um método de rastreamento baseado numa representação em forma de grafos contendo informações, tais como cor, tamanho, forma de regiões, assim como a informação temporal definida a partir das regiões de interese extraídas em vários quadros consecutivos. O método foi aplicado para o rastreamento de jogadores de futebol e os resultados apresentados mostraram-se bastante eficientes.

A segmentação de objetos em ambientes não controlados como, por exemplo, em jogos de futebol, é um problema difícil, principalmente na presença de mudanças significativas de iluminação. Neste trabalho, foi abordado ainda o problema da extração de fundo de imagens em situações de grandes mudanças de iluminação. Uma imagem de fundo é usada na segmentação de seqüência de vídeo baseada na diferença de imagens. O método foi avaliado na segmentação de jogadores de futebol filmado durante um dia ensolarado com presença de algumas nuvens. Os resultados mostraram-se satisfatórios mesmo em casos de mudanças bruscas na intensidade das imagens.

Finalmente, foi desenvolvido um ambiente visual que permite a construção de forma fácil e rápida de diferentes tipos de algoritmos para o processamento de seqüência de

imagens e rastreamento de objetos. Uma das principais vantagens do ambiente desenvolvido é a possibilidade de se construir algoritmos iterativos, os quais são essenciais no processamento de imagens de vídeo.

Várias extensões visando a melhoria deste trabalho podem ser consideradas. Na análise cinemática de movimentos humanos, por exemplo, podem ser utilizadas as informações 3D da posição dos marcadores para melhorar o algoritmo de rastreamento. O rastreamento de cada marcador em 3D é possível a partir da reconstrução de sua posição quadro a quadro, usando as imagens de todas as câmeras empregadas na filmagem. Estes algoritmos podem ser otimizados e implementados dentro do ambiente visual desenvolvido para permitir o processamento em tempo real.

O método utilizado na extração do fundo deve ser melhorado no que se refere ao problema dos extremos (limites) do trecho de quadros selecionados, quando os objetos ficam parados ou se movimentam muito lentamente. É importante considerar ainda a possibilidade de se paralelizar a extração do fundo e o processo de segmentação, em situações em que se pretende trabalhar como múltiplos processadores.

Para melhorar o rastreamento automático dos jogadores em situações complexas, tais como escanteio e cobrança de falta, devem ser usadas câmeras adicionais com campo de visão mais fechado. As imagens capturadas por estas câmeras podem ser processadas e consideradas durante o rastreamento.

Geralmente, o tempo gasto na verificação da trajetória de cada jogador é relativamente alto comparado com o tempo de processamento. Neste caso, pode ser considerado um mecanismo que permita uma verificação automática de situações em que o jogador se encontra isolado ou em oclusões simples. Por exemplo, se não existem blobs próximos ao blob rastreado, não precisa carregar a imagem para visualizar. O rastreamento pode ser melhorado, ainda, excluindo-se caminhos já percorridos no grafo. Desta forma, para cada jogador rastreado diminuem-se os caminhos possíveis a serem considerados no rastreamento dos demais jogadores.

A calibração das câmeras é importante para o rastreamento e precisão dos dados e, portanto, métodos alternativos de calibração devem ser considerados, tomando-se, por exemplo, as linhas do campo em vez de pontos fixos como referência para esta calibração.

Um outro problema complexo refere-se à extração da semântica de um jogo cujo método de rastreamento pode auxiliar na detecção automática de seus eventos. Uma fusão dos dados provenientes de câmeras de TV e de câmeras fixas deve fornecer o auxílio complementar à caracterização desta semântica. Geralmente, as imagens transmitidas pela TV mostram apenas os eventos e portanto os objetos na imagem tem uma melhor definição. Neste contexto, pode-se tentar identificar a bola e o número do jogador.

Apêndice A

Exemplo de implementação de uma função para o ambiente visual

A função *CFilterHigh* a seguir é um filtro que transforma para 255 todas as intensidades maiores que um dado valor (**mLevel**) de uma imagem em níveis de cinza ou em cada banda R, G e B da imagem colorida. Para que uma caixa possa ser inserida na interface é necessária a implementação das funções constructor/destructor, execução e teste de dados de entrada. O construtor cria um objeto definindo o número de entradas e saídas, tipo de dados de entrada e saída assim como alguns parâmetros de inicialização. A função **Execute** lê os dados da entrada e realiza o processamento requerido. A função **IsReady** testa se os dados de entrada estão disponíveis para o processamento. A função adicional **Serialize** permite armazenar/carregar os parâmetros do toolbox quando um algoritmo precisa ser salvo ou carregado.

```
 ////////////////////////////////////////////////////
///// Construction/Destruction
CFilterHigh::CFilterHigh()
{
    mLevel=40;    //default parameter of filter
    mClassName=("CFilterHigh");
    mGlyp=new CBox("11");  ///toolbox of one input and one output
    mInputList1.AddEnd("Image"); // Add gray scale and color image data
    mInputList1.AddEnd("ColorImage"); // type for the input of this toolbox
    mOutputList1.AddEnd("Image"); /// Add gray scale and color image data
    mOutputList1.AddEnd("ColorImage");  // type for output of this toolbox
}
CFilterHigh::~CFilterHigh()
{
```

```
}
//////////////////////////////////////////////////////////////////
////// Execute the operation of filter for grey scale or RGB image
bool CFilterHigh::Execute()
{
   ////if the input/output data is a gray scale image
   if(mSelectIn1=="Image" && mSelectOut1=="Image" ){
       CDataImage * temp;
       temp=(CDataImage *)mInput1; //Get the data from Input 1
       temp->FilterHigh(mLevel);  /// execute the operation
       mOutput1=temp2;   // put the data to output
       mOutput1Ready=TRUE;  //set the data ready flag to true
   }
   //// if the input/output data is a RGB color image
   if(mSelectIn1=="ColorImage" && mSelectOut1=="ColorImage"){
       CDataColorImage * temp;
       temp=(CDataImage *)mInput1; //Get the data from Input 1
       temp->FilterHigh(mLevel); ///execute the operation
       mOutput1=temp2;   // put the data to output
       mOutput1Ready=TRUE;  //set the data ready flag to true
   }
   return true;
}
//// Test if the input data is ready    ////
bool CFilterHigh::IsReady()
{
   if(mInput1Ready) return TRUE;
   else return FALSE;
}
/////  Save/load the parameters         ////
void CFilterHigh::Serialize(CArchive &ar)
{
   if( ar.IsStoring() ){
      ar << mLevel;
   } else {
      ar >> mLevel;
   }
}
```

Bibliografia

[1] Y. I. Abdel-Aziz and H. M. Karara. Direct linear transformation from comparator coordinates into object-space coordinates. In *Proccedings of ASP/UI symposion on close-range photogametry*, pages 1–18, 1971.

[2] K. Akita. Image Sequence Analysis of Real World Human Motion. *Pattern Recognition*, 17(1):73–83, 1984.

[3] J. Allen, R. Butterly, M. Welch, and R. Wood. The physical and physiological value of 5-a-side soccer training to 11-a-side match play. *Journal of Human Movement Studies*, 34:1–11, 1998.

[4] Luciana M. Andrade. Analise de marcha: protocolo experimental a partir de variáveis cinemáticas e antropométricas. Tese de Mestrado, Faculdade de Educação Física - Departamento de Educação Motora, Universidade Estadual de Campinas, 2002.

[5] A. G. N. Araujo, L. M. Andrade, and R. M. L. Barros. Proposição de um modelo de representação dos membros superiorese e escápula durante a marcha humana. In *Anais do X Congresso Brasileiro de Biomêcanica*, volume 1, pages 379–382, 2003.

[6] D. H. Ballard and C. M. Brown. *Computer vision*. Prentice-Hall, Inc., 1982.

[7] J. Bangsbo, L. Norregaard, and F. thorso. Activity profile of competitions soccer. *Canadian Journal of Sport Science*, 16(2):110–116, 1991.

[8] R. M. Barros, R. Brenzikofer, N. J. Leite, and P. J. Figueroa. Desenvolvimento e avaliação de um sistema para análise cinemática de movimento humanos. *Revista Brasileira de Engenharia Biomédica*, 15(1-2):79–86, 1999.

[9] R. M. Barros, M. R. Leite, R. Brenzikofer, E. C. Filho, P. Figueroa, and J. Iwanowicz. Respiratory pattern changes in eldery yoga practitioners. *Journal of Human Movement Studies*, 44:387–400, 2003.

[10] S. Beucher and F. Meyer. The morphological approach to segmentation: The watershed transformation. In E.R Dougherty, editor, *Mathematical Morphology in Image Processing*, chapter 12, pages 433–481. Marcel Decker, Inc, New York, 1992.

[11] R. Brenzikofer, P. Depra, E. Lima, R. Barros, and P. Figueroa. Spinal kinematics in normal walking using geometric curvature. In *Proc. XVIII Congress of the International Society of Biomechanics*, pages 16–17, 2001.

[12] A. L. Pachêco C. M. A. Loule and R. M. L. Barros. Análise de volumes respiratórios por videogrametria. In *Anais do X Congresso Brasileiro de Biomêcanica*, volume 2, pages 128–131, 2003.

[13] A. Cavallaro and T. Ebrahimi. Video object extraction based on adaptive background and statistical change detection. In *Proc. SPIE Electonic Imaging - Visual Comunication and Image Processing*, pages 465–475, 2001.

[14] P. Cox, F. Giles, and T. Pietrzykovski. Prograph: A step towards liberating programming from textual condicioning. In *Proc. IEEE Workshop on Visual Languages*, 1989.

[15] J. Crosbie, R. Vachalathiti, and R. Smith. Pattern of spinal motion during walking. *Gait and Posture*, 5:6–12, 1997.

[16] Ricardo M. Leite de Barros. *Concepção e Implementação de um Sistema para Análise Cinematica de Movimentos Humanos*. Tese de Doutorado, Universidade Estadual de Campinas - Brasil, 1997.

[17] S. D'Ottavio and C. Tranquilii. rindimento di jogador di futbole. *Sd5 - Rivista di Cultura Sportiva*, 24, 1993.

[18] G. De Haan E. B. Bellers. *De-interlacing: A Key Technology for Scan Rate Conversion*. Elsevier, 2000.

[19] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *in Proc. European Conference on Computer Vision*, pages 751–767, 2000.

[20] W. Erdmann. Quantification of games - preliminary kinematic investigation in soccer. In *Science and Football II*, pages 165–174, 1991.

[21] L. R. M. Fernandes and R. M. L. Barros. Análise das variáveis angulares dos dedos da mão durante a presão palmar de força. In *Anais do X Congresso Brasileiro de Biomêcanica*, volume 2, pages 46–49, 2003.

[22] G. Ferrigno, N. A. Borghese, and A. Pedotti. Pattern recognition in 3D automatic human motion analysis. *IPRS Journal of Photogrammetry and Remote Sensing*, 45(4):227–246, 8 1990.

[23] G. Ferrigno and A. Pedotti. Elite: A digital dedicated hardware system for movement analysis via real-time TV signal processing. *IEEE transaction on Biomedical Engineering*, 32(11):943–949, 1985.

[24] P. Figueroa, N. Leite, and R. Barros. A flexible software for tracking markers for human motion analysis. *Computer Methods and Programs in Biomedicine*, 72:155–165, 2003.

[25] P. Figueroa, N. J. Leite, R. M. Barros, and R. Brenzikofer. Tracking marker for human motion analysis. In *Proc. of IX European signal processing conference*, pages 941–944, 1998.

[26] Pascual. J. Figueroa. Perseguição de Marcadores para Análise de Movimentos Humanos. Tese de Mestrado, Universidade Estadual de Campinas, 1998.

[27] A. François and G. Medioni. Adaptive color background modeling for real-time segmentation of video streams. In *Proc. Int. Conf. on Image Science, System and Technology*, pages 227–232, 1999.

[28] D.M. Gavrila and L.S. Davis. Tracking of human in action: a 3D model-based approach. Technical report, Computer Vision Laboratory, CfAR, University of Maryland, 1996.

[29] Michael Grimand. A new measure of contrast: Dynamics. In *Proc. Image Algebra and Morphological Image Processing III*, pages 294–305, 1992.

[30] R. M. Haralick, S. R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *IEEE Trans.Pattern Analysis and Machine Intelligence*, 9(4):532–549, 7 1987.

[31] I. Haritaoglu, D. Harwood, and L. Davis. W^4: Real-time surveillance of people and their activities. *IEEE Trans.Pattern Analysis and Machine Intelligence*, 22(8):809–830, 2000.

[32] Simon Haukin. *Modern Filters*. Macmillan Publishing Company, 1989.

[33] E. Hennig and R. Briehle. Game analysis by GPS satellite tracking of soccer players. In *Proc. XI congress of Canadian Society of Biomechanics*, page 44, 2000.

[34] R. Hesel. Cutting your test development time with hp vee. In *HP professional books*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[35] S. S. Intille and A. F. Bobick. Visual tracking using closed-worlds. Technical report, M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 294, 1994.

[36] S. Iwase and H. Saito. Tracking soccer players based on homography among multiple views. In *Proceeding of SPIE*, pages 283–292, 2003.

[37] G. Jaffré and A. Crouzil. Non-rigid object localization from color model using mean shift. In *Proc. Int. Conf. Image Processing*, volume 3, pages 317–320, 2003.

[38] T. Josefsson, E. Nordh, and P. Eriksson. A flexible high-precision video system for digital recording of motor acts through lightweight reflex markers. *Computer methods and programs in biomedicine*, 49:119–129, 1996.

[39] T. Kawashiwa, K. Yoshino, and Y. Aok. A qualitative image analysis of group behaviour. In *Proc. Computer Vision and Pattern Recognition*, pages 690–693, 1994.

[40] T. Kimura. A visual language for keyboardless programming. Technical Report Technical ReportWUCS-86-6, Departament of Computer Science, Washington University, St. Louis, Misuri 63130, 1986.

[41] D. Koelma and A. Smeulders. A visual programming interface for an image processing environment. *Pattern Recognition Letters*, 6:1099–1109, 1994.

[42] J. Krist, M. Melluish, L. Kehl, and D. Crouch. Technical description of the optotrack 3D motion measurement system. In *Gangbildanalyse*, pages 23–39. Mecke Druck und Verlag, Duderstad, 1990.

[43] M.K. Leung and Y.H. Yang. First Sight: A human body outline labeling system. *IEEE Trans.Pattern Analysis and Machine Intelligence*, 17(4):359–377, 1995.

[44] V. Macellari. Costel - a computer peripheral remote sensing device for 3-dimentional monitoring of human motion. *J. Medical and Biological Engineering and Computer*, 21(3):311–318, 1983.

[45] K. Matsui, M. Iwase, M. Agata, T. Tanaka, and N. Ohnishi. Soccer image sequence computed by virtual camera. In *Proc. IEEE Conf. on Computer Vision and Pattern*, pages 23–25, 1998.

[46] S. R. Mayhew and H. A. Wenger. Time-motion analysis of professional soccer. *Journal of Human Movements Studies*, 11:49–52, 1985.

[47] S. J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler. Tracking groups of people. *Computer Vision and Image Understanding*, 80:42–56, 2000.

[48] R. Menezes, C. Dechechi, M. Misuta, P. Figueroa, and R. Barros. Analise de trajetoria, distancias percorridas e velocidades de jogadores de futebol durante uma partida. In *XXVI Simpósio Internacional de Ciência do Esporte, edição especial da revista brasileira de Ciência e Movimento*, page 116, 2003.

[49] M. Misuta, P. Figueroa, C. Dechechi, and R. Menezes. Validação do método de analise automatica de deslocamento de jogadores de futebol por videogrametria. In *Anais do X congresso brasileiro de Biomecanica*, pages 411–414, 2003.

[50] M. S. Misuta, T. G. Russomanno, L. B. Santis, D. D. Salgado, and R. M. L. Barros. Estudo da variabilidade de curvas de flexão do joelho no salto vertical. In *Anais do VIII Congresso Brasileiro de Biomêcanica*, pages 209–212, 1999.

[51] M. Mosconi and M. Porta. Iteration constructs in data-flow visual languages. *Computer Languages*, 26:67–104, 2000.

[52] C. J. Needman and R. D. Boyle. Tracking multiple sport players through occlusion, congestion and scale. *British Machine Vision Conference*, pages 93–102, 2001.

[53] S. A. Niyogi and E. H. Adelson. Analiyzing and Recognizing Walking Figure in XYT. Technical Report TR94-223, Perceptual Computing Section MIT Media Laboratory, 1992.

[54] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Prentice Hall, 1982.

[55] A. Pedotti, C. Farigo, and R. Rodano. Optimization of motor co-ordination in sport, an analytical approach. In *Biomechanics and Performance in Sports*, pages 145–160. Baumann and Schnordorf Eds, 1983.

[56] J. Rasure and C. S. Williams. An integrated data-flow language and software development environment. *Journal of Visual Languages and Computing*, 2:217–246, 1991.

[57] T. Reilly and V. Thomas. A motion analysis of work-rate in different positional roles in professional football match play. *Journal of Human Movement Studies*, 2:87–97, 1976.

[58] J. G. Richards. The measurement of human motion: A comparison of commercially available systems. *Human movement science*, 18:589–602, 1999.

[59] C. Ridder, O. Munkel, and H. Kirchner. Adaptive background estimation and foreground detection using kalman-filter. Technical report, Bavarian Research Center for Knowledge-Based Systems, 1995.

[60] K. Rohr. Towards Model-Based Recognition of Human Movements in Image Sequences. *CVGIP: Image Understanding*, 59(1):94–115, 1994.

[61] Karine. J. Sarro. Metodologia para análise da movimentação da caixa toráxica durante a respiração. Tese de Mestrado, Faculdade de Educação Física - Departamento de Educação Motora, Universidade Estadual de Campinas, 2003.

[62] A. Schürr. Bdl - a nondeterministic data flow programming languages with backtracking. In *Proc. IEEE Conf. on Visual Languages (VL'97)*, 1997.

[63] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, second printing, London, 1992.

[64] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision.* Chapman & Hall Computing, 1993.

[65] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[66] E. Stüssi and R. Müller. Vergleichende bewertung kommerziell erhälticher 3D-kinematik-systeme für die gangbildanalyse. In U. Boenick, M. Näder, and C. Mainka, editors, *Gangbildanalyse, Stand der Messtechnik und Bedeutung für die Orthopädie-Technick*, pages 86–97. Technische Universität Berlin, Mecke Druck und Verlag, Duderstad, 2 1990.

[67] R. Rivest T. Cormen, C. Leiserson. *Introduction to algorithms.* MIT Press, 1990.

[68] T. Taki, J. Hasegawa, and T. Fukumura. development of motion analysis system for quantitative evaluation of team work in soccer game. In *Proc. Int. Conf. Image Processing*, pages 815–818, 1996.

[69] N. Vandenbroucke, L. Macaire, and J. G. Postaire. Color pixel classification in an color hybrid color space. In *Proc. Int. conf. Image Processing*, pages 176–180, 1998.

[70] N. Vandenbroucke, L. Macaire, and J. G. Postaire. Color image segmentation by supervised pixel classificationin color texture feature space. application to soccer image segmentation. In *Proc. Int. Conf. Pattern Recognition*, volume 3, pages 625–628, 2000.

[71] G. M. Vose and G. Williams. Labview: laboratory virtual instrument engineering workbench. *BYTE*, 11(9):84–92, 1986.

[72] R. T. Withers, Z. Maricic, S. Wasilewski, and Kelly. Match analyses of australian professional soccer players. *Journal of Human Moviment Studies*, 8:159–176, 1982.

[73] H. Kim Y. Seo, S. Choi and K.S. Hong. Where are the ball and players? soccer game analysis with color-based tracking and image mosaick. In *Proc. International Conference on Image Analysis and Processing*, pages 196–203, 1997.

[74] M. Yeasin and S. Chaudhuri. Development of an automated images processing system for kinematic analisys of human motion. *Real-Time Imaging*, 6:55–67, 2000.