DISSERTAÇÃO DE MESTRADO

UMA PROPOSTA PARA A CLASSIFICAÇÃO DE AÇÕES HUMANAS BASEADA NAS CARACTERÍSTICAS DO MOVIMENTO E EM REDES NEURAIS ARTIFICIAIS

Thiago da Rocha

Brasília, fevereiro de 2012

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

UMA PROPOSTA PARA A CLASSIFICAÇÃO DE AÇÕES HUMANAS BASEADA NAS CARACTERÍSTICAS DO MOVIMENTO E EM REDES NEURAIS ARTIFICIAIS

Thiago da Rocha

Orientador: Prof. Dr. Alexandre Ricardo Soares Romariz

Co-orientador: Prof. Dr. Flávio de Barros Vidal

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA DE SISTEMAS ELETRÔNICOS E DE AUTOMAÇÃO

PUBLICAÇÃO: PPGEA.DM - 465/2012

Brasília, fevereiro de 2012

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

DISSERTAÇÃO DE MESTRADO

UMA PROPOSTA PARA A CLASSIFICAÇÃO DE AÇÕES HUMANAS BASEADA NAS CARACTERÍSTICAS DO MOVIMENTO E EM REDES NEURAIS ARTIFICIAIS

Thiago da Rocha

Relatório submetido ao Departamento de Engenharia

Elétrica como requisito parcial para obtenção

do grau de Mestre em Engenharia de Sistemas Eletrônicos e de Automação

Banca Examinadora

Prof. Dr. Alexandre Ricardo Soares Romariz, ENE/UnB _____
Orientador

Prof. Dr. Díbio Leandro Borges, CIC/UnB _____
Examinador interno

Prof. Dr. Luis Pereira Calôba, COPPE/UFRJ _____
Examinador externo

FICHA CATALOGRÁFICA

Rocha, Thiago da.
R672p Uma proposta para a classificação de ações humanas baseada nas características do movimento e em redes neurais artificiais Thiago da Rocha. – 2012. vii , 83f. :il.; 30 cm.

Dissertação (mestrado) - Universidade de Brasília, Faculdade de Tecnologia, Departamento de Engenharia Elétrica, Programa de Pós-Graduação em Engenharia de Sistemas Eletrônicos e de Automação, 2012.
Inclui bibliografia.
Orientação: Alexandre Ricardo Soares Romariz.

1. Redes Neurais (Computação). 2. Inteligência Computacional
3. Vigilância Eletrônica 4. Interação Homem-Máquina
5. Reconhecimento de Padrões 6. Visão por computador.
I. Romariz, Alexandre Ricardo Soares. II. Título.
CDU 621.397

REFERÊNCIA BIBLIOGRÁFICA

ROCHA, T. (2012). Uma proposta para a classificação de ações humanas baseada nas características do movimento e em redes neurais artificiais. Dissertação de Mestrado em Engenharia de Sistemas Eletrônicos e de Automação, Publicação PGEA.DM-465/2012, Departamento de Engenharia Elétrica, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 83p.

CESSÃO DE DIREITOS

AUTOR: Thiago da Rocha.
TÍTULO: Uma proposta para a classificação de ações humanas baseada nas características do movimento e em redes neurais artificiais.
GRAU: Mestre ANO: 2012

Thiago da Rocha
QS 5 Rua 310 Lote 14 Apto 204, Águas Claras
71964-360, Brasília, DF, Brasil.

Dedicatória

À minha mãe e à minha avó,
que me proporcionaram condições para que eu pudesse chegar até aqui.

Thiago da Rocha

Agradecimentos

Agradeço primeiramente à Deus, porque àqueles que são bons diante dEle, Deus dá sabedoria, conhecimento e felicidade.

Também agradeço aos meus professores e orientadores Prof. Romariz e Prof. Flávio, pela orientação, pelo ensinamento, pelas diretrizes, pela compreensão, pelo apoio, enfim, por todo convívio que tivemos nestes últimos dois anos e meio.

Aos professores Lee e Ladeira por me apresentarem outras técnicas de IA e me aprofundarem na disciplina. À professora Myléne por desvendar os segredos do processamento das imagens. À professora Sara, da minha graduação, através da qual aprendi boa parte do que sei sobre RNA.

Pela amizade dos meus amigos e amigas de Brasília e de São Paulo. Em especial à Adriana, ao Anderson, ao André, à Bárbara, ao César, ao Clóvis, ao Felipe, à Luciana, ao Jac, ao Rodrigo e à Vivian, pela ótima recepção em Brasília e também pelos momentos incomparáveis que vivemos. Ao Alexandre, ao Bruno, à Claudete, à Kelly, ao Hélio, ao Rogério e a Sandra Yara, pelas ótimas acolhidas em São Paulo. À Bete e ao Jorge, pelo show do U2. Novamente ao César e à Luciana, simplesmente por Miami e Orlando. Sem estes amigos eu não teria dado a pausa necessária para a produção dos ótimos resultados aqui apresentados. Abençoados são os que possuem amigos, os que os têm sem pedir. Porque amigo não se pede, não se compra, nem se vende. Amigo a gente sente!

Também agradeço aos meus chefes na CAIXA, Jair, Mourão, Vilson e Humberto, os quais entenderam meus horários malucos.

Ao LISA/CIC/UnB (Laboratório de Imagens, Sinais e Áudio) e aos seus coordenadores pelo espaço cedido. Neste lugar tive muitos momentos de inspiração. Também agradeço ao Prof. Queiroz pelo monstro verde.

Enfim, agradeço à todos que me apoiaram direta ou indiretamente na conclusão deste trabalho.

Thiago da Rocha

RESUMO

A recente revolução tecnológica ocorrida nas últimas décadas nos proporcionou a disponibilização de computadores com grande capacidade de armazenamento e processamento. Além disso, também temos ao nosso alcance câmeras de vídeo com alta qualidade de captura de imagens. Este cenário nos permite criar, armazenar e distribuir grande quantidade de vídeos. Diversas áreas da sociedade, tais como, vigilância, controle de tráfego e entretenimento, tem demandado o desenvolvimento de novas técnicas e metodologias automatizadas de análise de vídeos, as quais são independentes da avaliação humana ou de buscas exaustivas pelos arquivos de vídeo. Aplicações naturais para estas áreas podem incluir: reconhecimento baseado em movimento, navegação veicular, vigilância automatizada, monitoramento de fluxo de veículos e pedestres, controle de qualidade em fábricas, indexação de vídeos e iteração homem-máquina.

Neste trabalho propomos uma metodologia para o reconhecimento de ações humanas executadas em sequências de imagens usando Visão Computacional e Inteligência Computacional. Na etapa de Visão Computacional utilizamos uma combinação de duas técnicas de análise de movimento: Histograma de Fluxo Óptico Orientado e Análise de Contorno de Objetos. Na etapa de Inteligência Computacional nós utilizamos um Mapa-Auto Organizável (SOM, do inglês *Self-Organizing Map*) otimizado através da rede de Aprendizado por Quantização Vetorial (LVQ, do inglês *Learning Vector Quantization*).

Testamos a metodologia proposta com uma base de dados que contém diferentes tipos de ações humanas. Por meio dos resultados obtidos e comparando-os com outras propostas encontradas na literatura, demonstramos a utilidade e a robustez da técnica.

ABSTRACT

The technology evolution that we experienced over the last decades increased the availability of computers with high processing and storage capacity, and video cameras with high quality image capture. It made it easier to create, store and upload videos. Considering this scenario, the areas such as surveillance, traffic control and entertainment deal with increasingly high amounts of video information, and require the development of new methodologies and techniques for video analysis. The increase in the overall amount of available video has set a requirement for simpler video analysis, independent of human evaluation and exhaustive searches. Natural applications of automatic video analysis include: motion based recognition, vehicle navigation, surveillance automation, pedestrian and vehicle flow monitoring, quality control in factories, video indexing and man-machine interaction.

In this work we develop and test a method for recognition of human actions in sequence of images using Computer Vision and Computational Intelligence. The Computer Vision stage is a combination of two motion analysis techniques: Histogram of Oriented Optical Flow and Object Contour Analysis. For the Computational Intelligence stage we use a Self-Organizing Map (SOM) optimized through Learning Vector Quantization (LVQ).

We test the proposed method against a database with different kinds of human actions. From the results and comparing it to other proposals in the literature, we show the usefulness and robustness of this method.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

A evolução tecnológica que vivenciamos nas últimas décadas permitiu a proliferação de computadores com grande capacidade de processamento e armazenamento e de câmeras de vídeos com alta qualidade de captura de imagens. Isso fez com que nossa sociedade fosse capaz de gerar, armazenar e transmitir conjuntos maciços de vídeos. Esta capacidade vem aumentando à medida que os custos financeiros associados aos equipamentos de geração de vídeos estão reduzindo e à medida que eles têm se tornado portáteis e de simples manipulação. Também é relativamente fácil e barato armazenar e transmitir estes vídeos, porém sua análise e extração de informações não é uma tarefa trivial, exigindo muitas vezes a busca exaustiva pelo vídeo e a dependência de avaliação humana, aumentando os custos financeiros e temporais associados à análise destes vídeos.

As áreas de vigilância, trânsito, multimídia, entretenimento, esporte e biomédica são grandes geradores de dados em vídeos e tem potencial de aplicações que demandam o desenvolvimento de novas metodologias e técnicas para a análise de vídeos. Yilmaz *et al.* [14] citam alguns exemplos de aplicações:

- reconhecimento baseado em movimento, que permite executar o reconhecimento humano através de sua caminhada ou a detecção automatizada de objetos.
- vigilância automatizada, para o monitoramento de cenas e identificação de atividades suspeitas ou eventos inadequados.
- indexação de vídeos, para a recuperação e anotação automática de vídeos em banco de dados multimídia.
- interação homem-máquina, para o reconhecimento de gestos ou rastreamento do olhos para entrada de dados no computador.
- monitoramento de tráfego, para a coleta em tempo real de dados estatísticos do tráfego, e permitir um melhor controle do fluxo.
- navegação de veículos, para o planejamento de rotas baseado em vídeo e desvio de obstáculos.

Com o processamento de vídeos, que neste contexto são considerados apenas uma sequência de imagens, e a geração de padrões que representem a ação executada, podemos implementar algoritmos de reconhecimento de padrões que identifiquem o tipo de ação humana executada, fornecendo informações relevantes que servirão como parâmetros de entrada para outros processos automatizados, reduzindo os custos associados à análise manual das sequências de imagens.

Este trabalho tem como objetivo principal propor uma nova metodologia para o reconhecimento de ações humanas em sequências de imagens, baseada em Visão e Inteligência Computacional. Tal metodologia será descrita nos próximos Capítulos. Definimos ações humanas como sendo os movimentos executados por seres humanos e consideramos que todo o corpo está sendo capturado e representado nos

quadros da sequência de imagens. Na próxima Seção descrevemos brevemente uma relação não exaustiva de trabalhos publicados recentemente que tinham objetivos de pesquisa semelhantes ao deste trabalho.

1.1 TRABALHOS RELACIONADOS

Diversos trabalhos foram desenvolvidos visando propor metodologias para o reconhecimento de ações humanas. Em geral, busca-se definir uma forma de extração e representação das características do movimento e do objeto na sequência de imagens, gerando padrões de representação, os quais são utilizados por algoritmos de classificação de padrões para executar o reconhecimento da ação que está sendo efetuada na sequência de imagens. Nesta Seção descrevemos brevemente alguns trabalhos recentemente publicados que tratam esta problemática. Todos eles utilizaram o conjunto de vídeos disponibilizados pelo Laboratório de Visão Computacional do Instituto de Ciências Weizmann [12], a qual contém sequências de imagens onde nove pessoas diferentes executam diversas ações naturais, tais como, abaixar, correr, pular, entre outras. Este conjunto de vídeos está apresentado em detalhes na Seção 6.1. Como neste trabalho, as propostas encontradas na literatura utilizam técnicas de Visão Computacional para gerar representações dos movimentos e de Inteligência Computacional para gerar modelos classificadores dos movimentos.

Wang, Huang e Tan [15] utilizaram uma representação de movimento compacta para o reconhecimento de ações humanas. Após calcular o Fluxo Óptico das sequências de imagens por meio do algoritmo de Lucas e Kanade [16, 17], descrito na Seção 2.2.3, os autores particionam o Fluxo Óptico em $N \times N$ blocos, para gerar informações locais de fluxo óptico, as quais são integradas para gerar a informação global. Para cada bloco é gerado um histograma normalizado do fluxo, com 8 *bins*. A ideia principal por trás deste processo é a conversão do Fluxo Óptico das coordenadas cartesianas para coordenadas polares. Segundo os autores, a representação de um movimento na forma de velocidade e direção é mais natural do que a representação na forma de velocidade horizontal e velocidade vertical. Através dos histogramas gerados, representa-se um movimento pela suas principais direções (os *bins* do histograma) e pela velocidade (ou magnitude) dos vetores (a altura dos *bins* nos histogramas). Além disso, eles calculam algumas características estatísticas dos vetores dos fluxos ópticos, da forma e da trajetória, as quais são integradas para gerar a representação final do movimento. Esta representação é utilizada para gerar classificadores AdaBoost multi-classe. Para a base de dados de Weizmann, os autores alcançaram 93.3% de acerto na execução da tarefa de classificação.

O trabalho apresentado por Wang e Leckie [18] utiliza as informações derivadas das silhuetas espaço temporais para representar as ações humanas. O cálculo das silhuetas entre os quadros é efetuado através de métodos de extração de fundo e diferença temporal ou por rastreadores de contorno. Como o tamanho e a posição das silhuetas variam à medida que o movimento ocorre, a silhueta é centralizada, para que seja calculada uma média da silhueta. Para criar uma biblioteca de silhuetas médias, os autores quantizaram as silhuetas médias utilizando o algoritmo k-*means*, gerando grupos de silhuetas médias. Com esta proposta, eles alcançaram 96.8 % de acerto.

Ali *et al.*[19] utilizam os conceitos da teoria dos sistemas caóticos para modelar e analisar as dinâmicas não lineares das ações humanas. As trajetórias de diversos pontos de referência são agrupadas para repre-

sentação do sistema dinâmico não linear. Cada trajetória é então utilizada para reconstruir um espaço de fase, empregando um esquema de atraso. As propriedades deste espaço de fase são capturadas de forma a gerar um vetor de características. Utilizando o algoritmo do vizinho mais próximo, os autores alcançaram 92.6% de acerto.

Já Niebles e Li [20] utilizam uma coleção de características espaciais e espaço temporais para extrair estatísticas e a dinâmica dos pontos de interesse, as quais são combinadas para compor uma bolsa de características (*bag-of-features*). Nesta proposta as características são combinadas para executar a tarefa de classificação. Entre as características utilizadas podemos citar o mapa de bordas e os pontos de interesse. Para cada exemplo, cada característica irá definir (ou votar) a classe que o exemplo pertence. A classe mais votada é a escolhida para o exemplo apresentado. Utilizando o algoritmo SVM os autores alcançaram acerto de 72.8%.

Jhuang *et al.* [21] apresentam uma extensão de [22], em que uma solução biologicamente motivada executa uma modelagem estatística da cena. Jhuang *et al.* [21] alteraram as características da forma pelas características do movimento. O sistema proposto consiste de uma hierarquia de detectores de características espaço temporais. À medida que avançamos nos níveis hierárquicos do sistema, a complexidade dos detectores de características vai aumentando. Como entrada do sistema, uma sequência de imagens é analisada por um *array* de unidades sensíveis a direção, as quais, através dos estágios da hierarquia do sistema, se tornam detectores de características espaço temporais invariantes a posição. Eles extraem as informações de movimento local com conjunto de filtros de fluxo. As respostas obtidas são armazenadas localmente e convertidas para respostas de alto nível que serão comparadas a *templates* mais complexos aprendidos dos exemplos. Diferentes tipos de detectores de características foram utilizados, entre eles podemos citar os gradientes espaço temporais, Fluxo Óptico, e orientação espaço temporal. Na fase de classificação, os autores utilizaram o SVM e alcançaram diversos resultados. Para a base de Weizmann, alcançaram 98.8 % com o detector baseado em gradiente.

Thurau e Hlavac [23] utilizam um método para o reconhecimento de ações humanas baseado em poses primitivas. Eles estenderam um descritor baseado em Histogramas de Gradiente Orientado. As classes de ação são representadas pelos histogramas das poses primitivas. Estes histogramas são complementados com informações temporais, para representar a sequência de imagens. A classificação se dá pela comparação destes histogramas, executada através do algoritmo de vizinho mais próximo. Os autores alcançaram acerto de 94.4%.

Chaudhry *et al.* [4] propuseram que a representação de cada quadro da sequência de imagens fosse feita através de Histogramas de Fluxo Óptico Orientado (HOOF, do inglês *histogram of oriented optical flow*) e o reconhecimento de ações humanas fosse executado pela classificação das séries temporais dos HOOF. Utilizando as técnicas de vizinho mais próximo os autores alcançaram 94.4% de acerto.

1.2 ORGANIZAÇÃO DESTE TRABALHO

Este trabalho está organizado em sete Capítulos, incluindo esta Introdução. Ao final apresentamos as Referências Bibliográficas.

No Capítulo 2 descrevemos os conceitos e as definições comumente utilizados no campo de Visão Computacional. Além disso, apresentamos as definições de deslocamento óptico e Fluxo Óptico, necessários para a compreensão dos algoritmos de Cálculo de Fluxo Óptico propostos por Horn e Schunck [24], Lucas e Kanade [16, 17], Kanade, Lucas e Tomasini [10, 11], também descritos no Capítulo. Por fim, descrevemos a metodologia de cálculo dos Histogramas de Fluxo Óptico Orientado, que é utilizada para estimar uma distribuição dos vetores de Fluxo Óptico.

No Capítulo 3 apresentamos os métodos e conceitos utilizados para executar a detecção do contorno de partes de uma imagem, bem como detalhamos o algoritmo *Moore-Neighbor Tracing*, comumente utilizado para esta tarefa. Além disso, buscamos descrever as representações de contorno mais utilizadas e o processo de normalização deste contorno.

No Capítulo 4 descrevemos a teoria das Redes Neurais Artificiais (RNA), bem como as principais etapas envolvidas no desenvolvimento de aplicações que utilizam as RNA, suas principais arquiteturas e métodos de aprendizagem. Além disso, apresentamos detalhadamente dois de seus principais algoritmos, o SOM e a LVQ, ambos propostos por Teuvo Kohonen.

No Capítulo 5 apresentamos as etapas da metodologia proposta neste trabalho, que tem como objetivo executar o reconhecimento automatizado do tipo de ação humana que ocorreu em uma sequência de imagens. Apresentamos os parâmetros utilizados em cada etapa da metodologia e descrevemos suas entradas e saídas.

No Capítulo 6 descrevemos as características do conjunto de dados utilizados para validar a metodologia proposta. Também apresentamos os tipos de experimentos executados, bem como das técnicas de validação dos modelos de classificação gerados. Encerramos o Capítulo demonstrando os resultados encontrados e comparando-os com os resultados dos trabalhos para reconhecimento de ações humanas descritas na Seção 1.1.

No Capítulo 7 apresentamos as considerações finais do autor, as contribuições deste trabalho, as limitações e vantagens da metodologia proposta, bem como as sugestões de trabalhos futuros.

2 FLUXO ÓPTICO

Uma técnica de Visão Computacional comumente utilizada para avaliar o movimento entre dois quadros de uma sequência de imagens é o cálculo de seu respectivo Fluxo Óptico. Esta metodologia é executada sem nenhum conhecimento *a priori* acerca do conteúdo das imagens e é capaz de estimar o movimento que está ocorrendo na sequência de imagens. Neste Capítulo definimos os conceitos de imagem, sequência de imagens e Fluxo Óptico, bem como apresentamos os principais algoritmos para o cálculo do Fluxo Óptico e de sua representação baseada em Histogramas.

2.1 CONCEITOS E DEFINIÇÕES

2.1.1 Imagem e Sequência de imagens

Conforme apresentado por Gonzalez e Woods [25], podemos definir uma imagem como uma função bidimensional $f(x, y)$, onde x e y são coordenadas espaciais. A amplitude de f para qualquer par de coordenadas (x, y) é denominada a intensidade da imagem (ou o brilho) naquele ponto. Também é comum a utilização do termo nível de cinza para se referir a intensidade de imagens monocromáticas. Para convertermos uma imagem para o formato digital, suas coordenadas e amplitudes devem ser digitalizadas. A digitalização dos valores das coordenadas é conhecido como amostragem e a digitalização dos valores da amplitude como quantização. Quando tomamos quantidades discretas, com valores finitos para x, y e para a amplitude de f, podemos dizer que a imagem é uma imagem digital. No decorrer desta dissertação utilizaremos a seguinte representação para uma imagem digital:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.1}$$

onde M e N denotam, respectivamente, o número de linhas e colunas da imagem digital resultante da amostragem da imagem $f(x, y)$ e $M \times N$ e expressa o tamanho da imagem digital. Os elementos da matriz são chamados de pixels da imagem e possuem localizações, coordenadas (x, y), e valores particulares, amplitude de f em (x, y).

Uma sequência de imagens é uma função tridimensional $h(x, y, t)$ que possui uma ou mais imagens $f(x, y)$ tomadas em instantes de tempo discretos t.

2.1.2 Similaridade de regiões entre imagens, deslocamento óptico e Fluxo Óptico

Apresentamos a seguir uma adaptação da definição de correspondência óptica apresentada por Minetto [1], aqui denominada similaridade de regiões entre imagens. Considerando um pixel p de uma imagem digital J, podemos definir seu correspondente em uma imagem digital K como sendo o pixel q, tal que a similaridade entre os valores de J na vizinhança de p e de K na vizinhança de q são maximizados. Na Figura 2.1 Minetto [1] ilustrou este conceito.

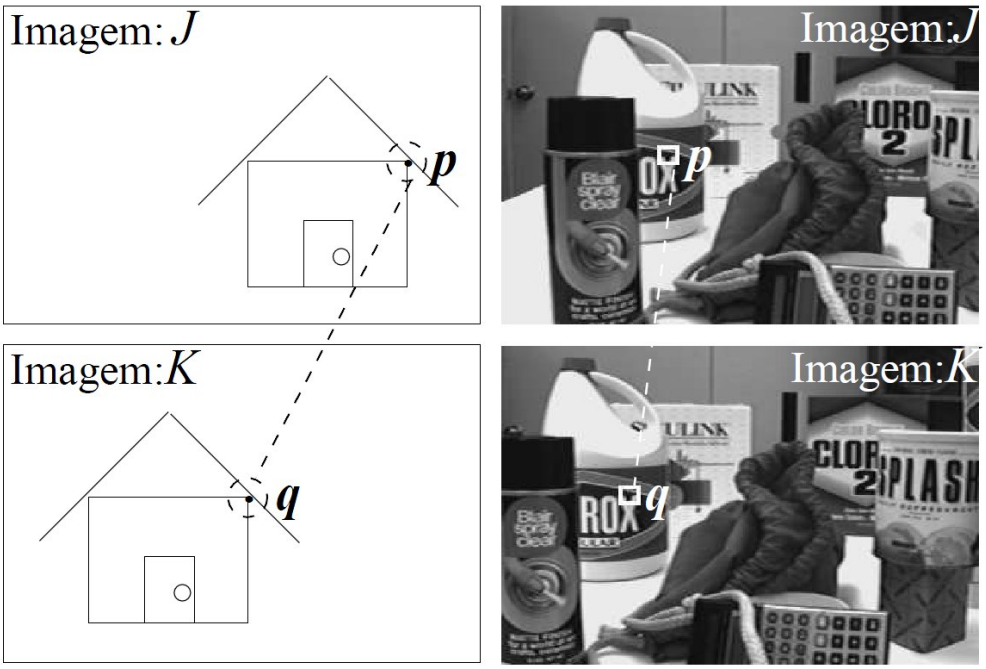


Figura 2.1: Similaridade de regiões entre imagens, ilustrada por Minetto [1]

Em geral, podemos aplicar este conceito a duas imagens que compõe uma sequência de imagens e foram obtidas da mesma cena em momento distintos, variando a posição dos objetos ou da câmera. Conforme Minetto [1], denotaremos o conceito de similaridade de regiões entre imagens por

$$J(p) \approx K(q). \tag{2.2}$$

Assumindo que q é similar a p, dizemos que o vetor $f = q - p$ é o deslocamento de p entre J e K, no domínio da imagem. Assim, definimos que o Fluxo Óptico de uma imagem J para uma imagem posterior K é uma função que mapeia cada ponto p ao seu vetor de deslocamento $f(p)$ e possui a seguinte propriedade:

$$J(p) \approx K(p + f(p)). \tag{2.3}$$

Na Figura 2.2 Minetto [1] apresenta uma ilustração de um Fluxo Óptico f, o qual foi amostrado em um conjunto fixo de pontos, $p_1, p_2, ..., p_n$, e produziu um conjunto de vetores $f_1, f_2, ..., f_n$. Cada vetor f_i é apresentado como um segmento de reta com origem em p_i, com a mesma direção de f_i e comprimento proporcional ao módulo de $f_i = |f_i|$.

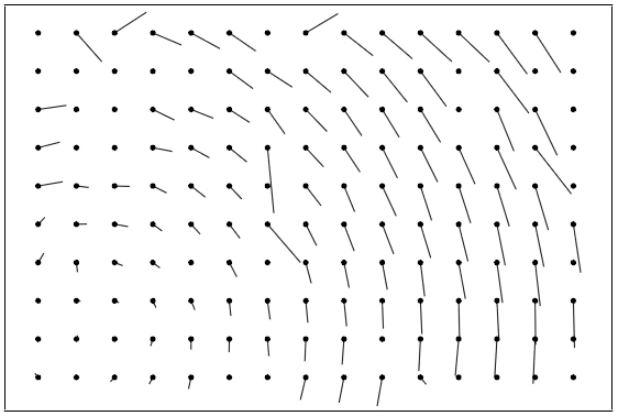


Figura 2.2: Representação de um Fluxo Óptico, ilustrada por Minetto [1]

O cálculo do Fluxo Óptico assume a hipótese de que a intensidade de uma região particular de uma imagem permaneça aproximadamente constante, à medida que a imagem vem sendo alterada com o tempo, considerando um curto período de tempo. Assim, assumimos que a intensidade de pixels correspondentes em uma sequência de imagens permanece constante. Beauchemin e Barron [2] e Minetto [1] apresentam uma definição formal para esta hipótese, apresentada na Equação 2.4.

Considerando que as imagens J e K compõe a sequência de imagens contínua H, então:

$$J(p) = H(p,t) \quad e \quad K(p) = H(p,t+\delta t) \tag{2.4}$$

onde δt corresponde a um intervalo de tempo pequeno. Desta forma, considerando x e y as coordenadas de p e f_x e f_y as componentes do deslocamento $f(p)$ entre os instantes t e $t+\delta t$, podemos reescrever a Equação 2.2 como

$$H(x,y,t) = H(x+\delta x, y+\delta y, t+\delta t) \tag{2.5}$$

onde δx e δy são os deslocamentos locais nas coordenadas x e y, respectivamente, na região da imagem (x,y,t) , depois de um tempo δt. Expandindo o lado direito da Equação 2.5 pela série de Taylor em relação a δx, δy e δt obtemos

$$H(x,y,t) = H(x,y,t) + \delta x\frac{\partial H}{\partial x}(x,y,t) + \delta y\frac{\partial H}{\partial y}(x,y,t) + \delta t\frac{\partial H}{\partial t}(x,y,t) + O(\delta t^2) \tag{2.6}$$

onde $\frac{\partial H}{\partial x}(x,y,t)$, $\frac{\partial H}{\partial y}(x,y,t)$ e $\frac{\partial H}{\partial t}(x,y,t)$ são as derivadas parciais em x,y e t e $O(\delta t^2)$ é o termo de segunda ordem, geralmente não considerado por ser desprezível. Subtraindo $H(x,y,t)$ em ambos os lados, desprezando $O(\delta t^2)$, omitindo (x,y,t) e dividindo a equação por δt podemos reescrever a Equação 2.6 de modo que:

$$\frac{\partial H}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial H}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial H}{\partial t} = 0 \tag{2.7}$$

Considerando $u = \frac{\delta x}{\delta t}$ e $v = \frac{\delta y}{\delta t}$ e utilizando as abreviações H_x, H_y e H_t para as derivadas parciais do brilho da imagem com respeito a x, y e t, respectivamente, obtemos uma equação linear simples com apenas duas variáveis desconhecidas, conforme Equação 2.8.

$$H_x u + H_y v + H_t = 0 \tag{2.8}$$

O vetor $\vec{v} = (u, v)$ representa o Fluxo Óptico diferencial em determinado ponto da imagem, ou, mais especificamente, é a velocidade instantânea de um pixel (x, y) no instante t. A Equação 2.8, conhecida como equação de restrição do Fluxo Óptico, pode ser reescrita como

$$(H_x, H_y).(u, v) = -H_t \tag{2.9}$$

onde (H_x, H_y) é o gradiente espacial de H no instante t e define uma restrição a velocidade $\vec{v}$, conforme ilustrado por Beauchemin e Barron [2] através da Figura 2.3.

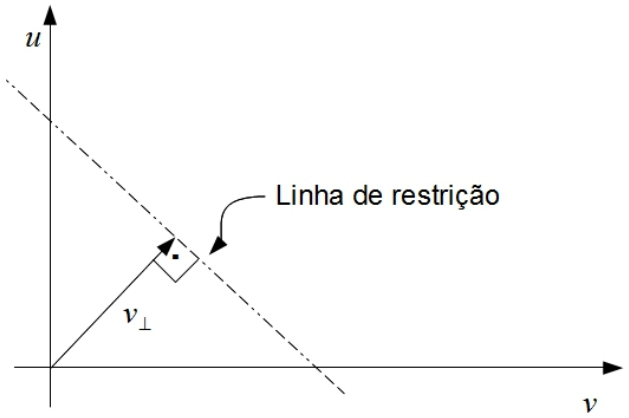


Figura 2.3: Equação de restrição do Fluxo Óptico define uma linha no espaço da velocidade [2].

Na Figura 2.3, o vetor de velocidade normal $v_\perp$ é definido como o vetor perpendicular a linha de restrição, que é a velocidade com a menor magnitude na linha de restrição do Fluxo Óptico. Com esta restrição não é possível calcularmos ambos os componentes de $\vec{v}$, considerando que o vetor possui dois componentes e apenas uma equação de restrição foi obtida, sendo possível estimar apenas o componente na direção do gradiente local da função de intensidade da imagem. Este problema é conhecido como problema da abertura do Fluxo Óptico. Segundo Beauchemin e Barron [2], somente em regiões da imagem onde existam informações de intensidade bem estruturadas é que o movimento pode ser totalmente estimado com o uso da equação de restrição do Fluxo Óptico.

Minetto [1] apresentou, como mostra a Figura 2.4, uma ilustração para a compreensão do problema da abertura, onde cada ponto p na primeira imagem pode possuir vários pontos q na segunda imagem com similaridades locais iguais ou muito próximas. A cena ilustra um quadrado de cor uniforme se movendo diagonalmente para cima, sendo que a posicional inicial está ilustrada com linha contínua e a final com linha tracejada. Para a determinação da similaridade local, da região mais similar e do Fluxo Óptico dos pontos utilizou-se as janelas circulares, indicadas com linha cheia na imagem de origem e com linha tracejada na imagem de destino. Podemos observar que para os pontos a e c a ambiguidade de similaridade

de regiões entre imagens se estende ao longo de uma linha, enquanto que para o ponto b o deslocamento é bem definido. Para o ponto d, a ambiguidade se estende sobre uma região.

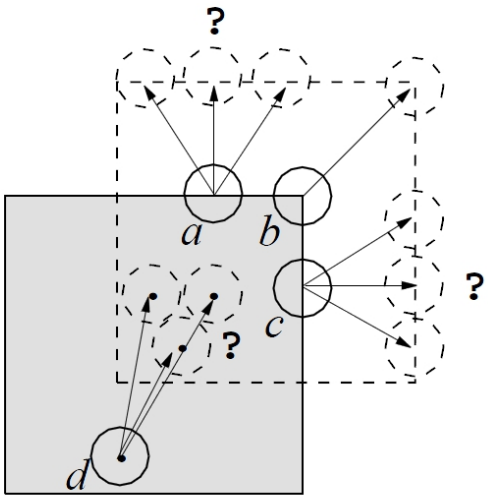


Figura 2.4: Problema da abertura, ilustrado por Minetto [1].

Assim, podemos estimar apenas $v_{\perp}$, o componente do movimento na direção do gradiente local da função de intensidade de uma imagem. Diversos algoritmos foram propostos com o objetivo de contornar este problema. Apresentamos na Seção 2.2 os algoritmos mais importantes e conhecidos, descrevendo suas principais características.

2.2 CÁLCULO DO FLUXO ÓPTICO

O cálculo do Fluxo Óptico consiste em associar um vetor de velocidade a cada pixel da imagem ou, equivalentemente, um deslocamento que representa a distância de um pixel que se moveu entre o quadro anterior e o atual.

Barron, Fleet e Beauchemin [26], Galvin *et al.* [27] e McCane *et al.* [28] apresentaram estudos de comparação do desempenho de diversas técnicas de Cálculo do Fluxo Óptico, as quais podem ser divididas em métodos diferenciais, baseado em regiões, baseado em energia e baseado em fases.

Apesar das diversas diferenças que podem ser encontradas entre os métodos de cálculo de Fluxo Óptico, Barron, Fleet e Beauchemin [26] afirmam que muitas destas técnicas podem ser vistas conceitualmente em três estágios de processamento:

1. pré filtragem, ou suavização, com filtros de passa-baixa/passa-alta para extração da estrutura do sinal de interesse e para alterar a taxa de sinais com ruído.
2. extração de medidas básicas, tais como derivadas espaço-temporal (para medir os componentes da

velocidade) ou correlação local das áreas.

3. integração destas medidas, para produzir um campo bidimensional do fluxo.

Na Seção seguinte descrevemos os conceitos relacionados com a estimativa das derivadas parciais de um ponto de imagem, que são fundamentais para a compreensão dos algoritmos de cálculo do Fluxo Óptico. Em seguida, apresentaremos o algoritmo proposto por Horn e Schunck [24], que busca atender a critérios de gradiente e suavidade global; o algoritmo de Lucas e Kanade [16, 17], que calcula o Fluxo Óptico diferencial; e o algoritmo de Kanade, Lucas e Tomasi [10, 11], que determina o Fluxo Óptico de maneira piramidal e com mais eficiência, também conhecido como algoritmo KLT.

2.2.1 Estimativa das Derivadas Parciais

A estimativa das derivadas parciais é um fator importante para o cálculo do Fluxo Óptico, devendo ser uma estimativa consistente. A estimativa das derivadas parciais é tomada a partir de um conjunto discreto de medidas de intensidades. Horn e Schunck [24] apresentaram uma equação que relaciona as alterações na intensidade de um ponto em uma imagem ao movimento do padrão de intensidade. Como definido anteriormente, $H(x, y, t)$ denota a intensidade de uma imagem no ponto (x, y) tomada no instante de tempo discreto t. Quando um padrão se move, a intensidade de um ponto particular permanecerá constante, de modo que

$$\frac{dH}{dt} = 0. \tag{2.10}$$

Aplicando a regra da cadeia podemos notar que

$$\frac{\partial H}{\partial x}\frac{dx}{dt} + \frac{\partial H}{\partial y}\frac{dy}{dt} + \frac{\partial H}{\partial t} = 0 \tag{2.11}$$

Considerando $u = \frac{dx}{dt}$, $v = \frac{dy}{dt}$, $H_x = \frac{\partial H}{\partial x}$, $H_y = \frac{\partial H}{\partial y}$, $H_t = \frac{\partial H}{\partial t}$, podemos reescrever a Equação 2.11,

$$H_x u + H_y v + H_t = 0, \tag{2.12}$$

de onde notamos que possuímos uma equação linear simples com duas variáveis desconhecidas u e v, que formam o vetor de movimento do ponto, ou o Fluxo Óptico. As derivadas parciais H_x, H_y e H_t podem ser aproximadas por vários métodos. Horn e Schunck [24] usam um conjunto que permite estimar H_x, H_y e H_t de um ponto no centro de um cubo formado por oito medidas, cuja relação espacial e temporal entre estas medidas está apresentada na Figura 2.5.

Cada uma das estimativas das derivadas parciais correspondem a média das diferenças das quatro medidas adjacentes no cubo, conforme apresentado nas Equações 2.13, 2.14 e 2.15.

$$H_x \approx \frac{(H_{i,j+1,k} - H_{i,j,k}) + (H_{i+1,j+1,k} - H_{i+1,j,k}) + (H_{i,j+1,k+1} - H_{i,j,k+1}) + (H_{i+1,j+1,k+1} - H_{i+1,j,k+1})}{4} \tag{2.13}$$

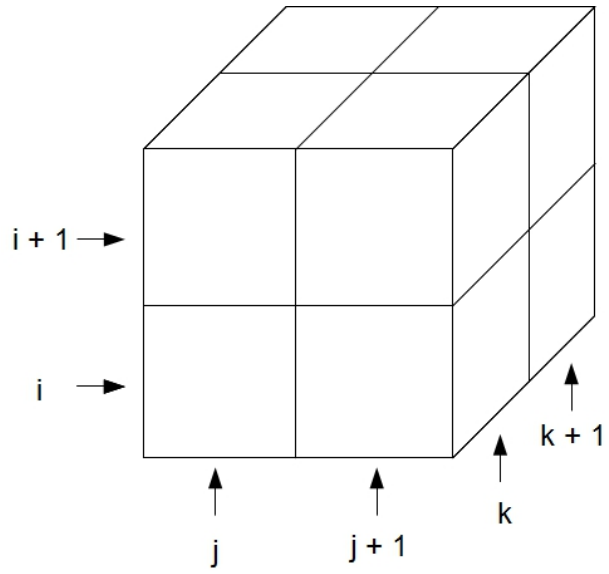


Figura 2.5: Relações espaciais e temporais, para a estimativa das derivadas parciais de um ponto da imagem, posicionado no centro do cubo, ilustrada por Horn e Schunck [24]. O índice da coluna j a eixo x na imagem, o índice da linha i corresponde a eixo y e o índice k no eixo do tempo.

$$H_y \approx \frac{(H_{i+1,j,k} - H_{i,j,k}) + (H_{i+1,j+1,k} - H_{i,j+1,k}) + (H_{i+1,j,k+1} - H_{i,j,k+1}) + (H_{i+1,j+1,k+1} - H_{i,j+1,k+1})}{4} \tag{2.14}$$

$$H_t \approx \frac{(H_{i,j,k+1} - H_{i,j,k}) + (H_{i+1,j,k+1} - H_{i+1,j,k}) + (H_{i,j+1,k+1} - H_{i,j+1,k}) + (H_{i+1,j+1,k+1} - H_{i+1,j+1,k})}{4} \tag{2.15}$$

2.2.2 O algoritmo de Horn e Schunck

O método proposto por Horn e Schunck desenvolvido em 1981 foi uma das primeiras técnicas a considerar que a intensidade dos pixels na sequência de imagens é constante. Além desta restrição, com o objetivo de obter os dois componentes do movimento $\vec{v}(u, v)$, uma segunda restrição foi introduzida para produzir um sistema totalmente determinado (duas equações, com duas variáveis desconhecidas). Horn e Schunck buscaram apresentar uma proposta que contornasse o problema da abertura assumindo a hipótese de que o Fluxo Óptico $v(x, y)$ varia suavemente com a posição (x, y). Em outras palavras, pixels vizinhos possuem movimento similar, produzindo um fluxo uniforme ou regular. Desta forma, o problema se resume a minimizar a energia total

$$\varepsilon^2_{total} = \int\int (\alpha^2 \varepsilon^2_{regularidade} + \varepsilon^2_{fluxo}) dxdy. \tag{2.16}$$

O termo $\varepsilon_{regularidade}$ é calculado através da Equação 2.17 e corresponde a medida de deslocamento, ou variação espacial, do Fluxo Óptico, o qual é ponderado pelo parâmetro α^2, que irá depender do erro

de quantização da imagem e do nível de ruído presente na sequência de imagens e determina a influência da restrição de regularidade na minimização. Em [24], os autores sugerem que a magnitude de α^2 seja aproximadamente proporcional ao ruído estimado em $H_x^2 + H_y^2$.

$$\varepsilon_{regularidade} = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{2.17}$$

O termo ε_{fluxo} é calculado através da Equação 2.18 e corresponde ao erro referente às alterações na taxa de intensidade na imagem, que violam a restrição do Fluxo Óptico expressa pela Equação 2.8.

$$\varepsilon_{fluxo} = H_x u + H_y v + H_t \tag{2.18}$$

O problema de minimização da energia total ε_{total}^2 pode ser reduzido a solucionar um conjunto de equações diferenciais, que são geradas por variações de cálculo, conforme apresentado pelas Equações 2.19 e 2.20.

$$H_x^2 u + H_x H_y v = \alpha^2 \nabla^2 u - H_x H_t \tag{2.19}$$

$$H_x H_y u + H_y^2 v = \alpha^2 \nabla^2 v - H_y H_t \tag{2.20}$$

Usando a aproximação do Laplaciano obtemos o sistema

$$\left(\alpha^2 + H_x^2\right) u + H_x H_y v = \left(\alpha^2 \bar{u} - H_x H_t\right) \tag{2.21}$$

$$H_x H_y u + \left(\alpha^2 + H_y^2\right) v = \left(\alpha^2 \bar{v} - H_y H_t\right) \tag{2.22}$$

O determinante da matriz de coeficientes é igual a $\alpha^2(\alpha^2 + H_x^2 + H_y^2)$. Resolvendo o sistema para u e v, obtemos a solução de 2.21 e 2.22, que é dada por

$$(\alpha^2 + H_x^2 + H_y^2) u = +(\alpha^2 + H_y^2)\bar{u} - H_x H_y \bar{v} - H_x H_t \tag{2.23}$$

$$(\alpha^2 + H_x^2 + H_y^2) v = -H_x H_y \bar{u} + (\alpha^2 + H_x^2)\bar{v} - H_y H_t \tag{2.24}$$

Estas equações podem ser reescritas alternativamente como

$$u = \bar{u} - \frac{H_x(H_x \bar{u} + H_y \bar{v} + H_t)}{\alpha^2 + H_x^2 + H_y^2} \tag{2.25}$$

$$v = \bar{v} - \frac{H_y(H_x \bar{u} + H_y \bar{v} + H_t)}{\alpha^2 + H_x^2 + H_y^2} \tag{2.26}$$

Com o objetivo de minimizar as Equações 2.25 e 2.26, Horn e Schunck [24] sugerem um processo de solução iterativa, conforme Equações 2.27 e 2.28.

$$u^{n+1} = \bar{u}^n - \frac{H_x(H_x\bar{u}^n + H_y\bar{v}^n + H_t)}{\alpha^2 + H_x^2 + H_y^2} \tag{2.27}$$

$$v^{n+1} = \bar{v}^n - \frac{H_y(H_x\bar{u}^n + H_y\bar{v}^n + H_t)}{\alpha^2 + H_x^2 + H_y^2} \tag{2.28}$$

onde $\bar{u}^n$ e $\bar{v}^n$ são as velocidades médias dos vizinhos de u e v na iteração n.

2.2.3 O algoritmo de Lucas e Kanade

Lucas e Kanade [16, 17] apresentaram outra proposta para resolver o problema da abertura do Fluxo Óptico. Segundo Bradski e Kaehler [3] o método de cálculo do Fluxo Óptico proposto por Lucas e Kanade [16, 17] assume três hipóteses:

1. *Constância da intensidade*, que assume que um pixel de objeto qualquer em uma cena de uma sequência de imagens não sofrerá alterações em sua aparência à medida que ele se move de quadro a quadro. Quando falamos de imagens em escala de cinza estamos nos referindo a intensidade do pixel, que não será alterada à medida que ele se desloca pelo plano da imagem.

2. *Persistência temporal (ou movimentos pequenos)*. O movimento da imagem é alterado lentamente à medida que o tempo passa, ou seja, os objetos nas imagens não se movem muito de um quadro para outro.

3. *Coerência espacial (ou rigidez no movimento)*. Pontos vizinhos numa cena, que pertencem a mesma superfície, apresentam movimentos similares.

A primeira restrição diz respeito à constância da intensidade. Os pixels de uma trajetória devem possuir a mesma aparência em todo o tempo, conforme apresentado na Equação 2.7.

Considerando $\frac{\delta x}{\delta t} = u$, $\frac{\delta y}{\delta t} = v$, $\frac{\partial H}{\partial x} = H_x$, $\frac{\partial H}{\partial y} = H_y$ e $\frac{\partial H}{\partial t} = H_t$ e isolando a derivara parcial de t, podemos reescrever esta equação de modo que

$$H_x u + H_y v = -H_t. \tag{2.29}$$

Considerando a última restrição apresentada por Bradski e Kaehler [3], que indica que uma região de pixels locais se movem com coerência, Lucas e Kanade [16, 17] definiram a noção de similaridade em uma vizinhança bidimensional, denominada janela de integração. Conforme apresentado por Bouguet [29], considerando que I e J são duas imagens digitais que compõe uma sequência de imagens H, que $p = [p_x \quad p_y]^T$ é um ponto da imagem I, $q = [q_x \quad q_y]^T$ é um ponto da imagem J, e que $\vec{v} = [u \quad v]^T$ é o vetor de velocidade do ponto p (também conhecido como Fluxo Óptico), a meta do algoritmo de cálculo do Fluxo Óptico é encontrar a localização de $q = p + v = [p_x + u \quad p_y + v]^T$ na imagem J, tal que

$I(p) \approx J(q)$. O vetor $\vec{v}$ é definido como o vetor que minimiza a função residual ϵ, apresentada na Equação 2.30.

$$\epsilon(\vec{v}) = \epsilon(u,v) = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} (I(x,y) - J(x+u, y+v))^2 \tag{2.30}$$

onde w_x e w_y são dois inteiros que definem o tamanho da janela de integração $W = (2w_x + 1) \times (2w_y + 1)$, onde será tomada a medida de similaridade $I(p) \approx J(q)$. Tipicamente w_x e w_y são definidos como 2,3,4,5,6, ou 7 pixels. Na Figura 2.6 ilustramos um exemplo de janela de integração com 25 pixels, onde $w_x = 2$ e $w_y = 2$.

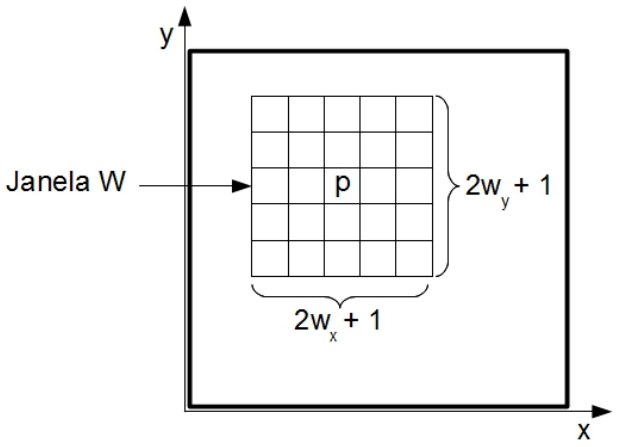


Figura 2.6: Exemplo de janela de integração, contendo 25 pixels, sendo $w_x = 2$ e $w_y = 2$.

Desta forma, Lucas e Kanade [16, 17] propuseram calcular o Fluxo Óptico de um pixel p utilizando o sistema de equações formado pelos pixels ao seu redor, internos a janela de integração W. Ao utilizarmos uma janela de pixels W, onde $2w_x + 1 = r$ e $2w_y + 1 = s$, poderemos calcular o movimento de p pelo seguinte conjunto de $n = rs$ equações:

$$\begin{array}{ccccc} H_{x1}u & + & H_{y1}v & = & -H_{t1} \\ H_{x2}u & + & H_{y2}v & = & -H_{t2} \\ H_{x3}u & + & H_{y3}v & = & -H_{t3} \\ \vdots & & \vdots & & \vdots \\ H_{xn}u & + & H_{yn}v & = & -H_{tn}, \end{array} \tag{2.31}$$

o qual pode ser reescrito em sua forma matricial de modo que

$$\begin{bmatrix} H_{x1} & H_{y1} \\ H_{x2} & H_{y2} \\ H_{x3} & H_{y3} \\ \vdots & \vdots \\ H_{xn} & H_{yn} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} H_{t1} \\ H_{t2} \\ H_{t3} \\ \vdots \\ H_{tn} \end{bmatrix} \tag{2.32}$$

ou

$$A\vec{v} = -b \tag{2.33}$$

Assim, obtemos um sistema com duas incógnitas, com mais de duas equações, tornando o sistema determinado em excesso. Podemos utilizar o método dos quadrados mínimos para encontrar uma solução com erro mínimo. Em sua forma padrão $\min\|Ad - b\|^2$ é resolvido do seguinte modo:

$$(A^T A)d = A^T(-b). \tag{2.34}$$

A partir desta relação, podemos calcular os componentes do Fluxo Óptico u e v:

$$\vec{v} = \begin{bmatrix} u \\ v \end{bmatrix} = (A^T A)^{-1} A^T(-b), \tag{2.35}$$

onde

$$A^T A = \begin{bmatrix} \sum H_x H_x & \sum H_x H_y \\ \sum H_x H_y & \sum H_y H_y \end{bmatrix} \tag{2.36}$$

e

$$A^T(-b) = -\begin{bmatrix} \sum H_x H_t \\ \sum H_y H_t \end{bmatrix}. \tag{2.37}$$

O problema poderá ser resolvido quando $A^T A$ possuir uma matriz inversa, ou seja, quando seu posto for completo, que ocorre quando a matriz possui dois autovetores grandes, que é equivalente a dizer que a sequência de imagens H possui gradiente significativo na vizinhança W do pixel que está sendo rastreado [1, 3]. Conforme exemplificado por Minetto [1], supondo que queremos rastrear o ponto p e que a matriz $A^T A$ possui autovalores $\lambda^{'}$ e $\lambda^{''}$ nulos, isso significa que a vizinhança do ponto p tem valor constante, e a matriz é singular, não permitindo o cálculo do Fluxo Óptico. Caso apenas um dos autovalores seja nulo, então o brilho da vizinhança varia apenas em uma única direção e a matriz será singular. Caso ambos sejam significativos, a matriz é invertível e o fluxo no ponto p é bem definido. Estas características das regiões de vizinhança influenciam diretamente no desempenho do algoritmo de Lucas e Kanade. A existência de detalhes na vizinhança permite a inversão da matriz $A^T A$ e o cálculo do Fluxo Óptico. A ausência de detalhes, com regiões de brilho constante ou que o brilho varia em apenas uma direção, tornam a matriz nula e impossibilitam o cálculo do Fluxo Óptico dos pontos naquela região.

2.2.4 O algoritmo de Kanade, Lucas e Tomasi (KLT)

Os dois algoritmos apresentados nas Seções anteriores consideram apenas movimentos pequenos. Caso os objetos estejam se movimentando rapidamente, os pixels se movimentarão muito rápido e as máscaras de cálculo das derivadas espaço temporais irão falhar. Segundo Bouguet [29], bons algoritmos de rastreamento

possuem dois componentes chaves: a acurácia e a robustez. A acurácia diz respeito principalmente à capacidade de incorporar as regiões de sub-pixels locais ao rastreamento e rastreá-las com altas taxas de precisão, especialmente em áreas da imagem onde ocorre a oclusão, que são locais com grande potencial de existência de dois ou mais grupos de pixels com diferentes fluxos de velocidades. Com o objetivo de alcançar uma boa acurácia tendemos a escolher janelas de integração pequenas, para mantermos os detalhes da imagem. A robustez diz respeito a sensibilidade do rastreamento a alterações nos níveis de intensidade da imagem e no tamanho do movimento na imagem. Intuitivamente preferimos janelas de integração maiores para movimentos maiores. Desta forma, a escolha do tamanho da janela de integração deverá considerar o nível de acurácia, para manter os detalhes, e o nível de robustez, para rastrear movimentos maiores, devendo manter um equilíbrio entre estes níveis.

Buscando estabelecer um modelo que apresentasse alta robustez, uma implementação piramidal do algoritmo clássico de Lucas e Kanade, o algoritmo de rastreamento por correspondências KLT, foi proposto por Lucas e Kanade [17] e desenvolvido por Tomasi e Kanade [10]. Posteriormente, buscando melhorar o desempenho do algoritmo, Shi e Tomasi o estenderam com a inclusão de um passo de seleção de boas características para rastrear. Em resumo, o algoritmo KLT identifica boas características para rastrear e em seguida retorna indicações de quão bem o rastreamento de cada ponto está ocorrendo. Antes de detalharmos o processo executado pelo algoritmo KLT, apresentaremos os conceitos envolvidos na representação pirâmidal imagens e na seleção de boas características, que são utilizados nas iterações do KLT. Ambos estão descritos tendo como base o que foi apresentado por Bouguet [29].

2.2.4.1 Representação piramidal de imagens

Uma pirâmide de imagens corresponde a um conjunto de imagens capturadas de uma mesma cena, que são calculadas de maneira recursiva a partir de uma única imagem original. Partindo de uma imagem inicial, que está no nível mais baixo da pirâmide, calculam-se as imagens dos níveis superiores, reduzindo-se a resolução das imagens à medida que o nível da pirâmide aumenta, conforme ilustrado na Figura 2.7. Formalmente, podemos considerar uma imagem genérica I de tamanho $n_x \times n_y$, a qual utilizaremos para gerar uma pirâmide de imagens. A imagem de nível zero é a imagem inicial, $I^0 = I$, que possui a maior resolução. A largura e altura desta imagem são definidas como $n_x^0 = n_x$ e $n_y^0 = n_y$, respectivamente. A pirâmide é construída recursivamente, de modo que calculamos I^1 a partir de I^0, I^2 de I^1, I^3 de I^2, e assim por diante. Denotamos por n_x^{L-1} e n_y^{L-1} a largura e altura de I^{L-1}, a qual é calculada como apresentado na Equação 2.38.

$$\begin{aligned} I^L(x,y) \quad = \quad & \tfrac{1}{4} I^{L-1}(2x, 2y) + \\ & \tfrac{1}{8}(I^{L-1}(2x-1, 2y) + I^{L-1}(2x+1, 2y) + I^{L-1}(2x, 2y-1) + I^{L-1}(2x, 2y+1)) + \\ & \tfrac{1}{16}(I^{L-1}(2x-1, 2y-1) + I^{L-1}(2x+1, 2y+1) + I^{L-1}(2x-1, 2y+1) + I^{L-1}(2x+1, 2y+1)) \end{aligned} \tag{2.38}$$

A largura n_x^L e a altura n_y^L de I^L são os maiores valores inteiros que atendem as condições apresentadas nas Equações 2.39 e 2.40.

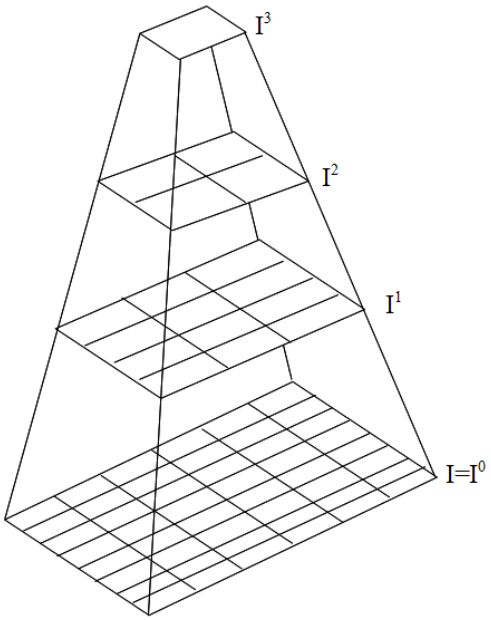


Figura 2.7: Exemplo de uma pirâmide de imagens, utilizada no algoritmo KLT, adaptado de [1].

$$n_x^L \leq \frac{n_x^{L-1} + 1}{2} \tag{2.39}$$

$$n_y^L \leq \frac{n_y^{L-1} + 1}{2} \tag{2.40}$$

Com estas restrições, para uma imagem I de tamanho 640×480, o tamanho das imagens I^1,I^2,I^3 e I^4 são respectivamente 320×240,160×120,80×60 e 40×30.

2.2.4.2 Seleção de boas características para rastrear

Conforme exemplificado por Bradski e Kaehler [3], se todos os pontos de uma imagem são idênticos ou muito similares, então um processo de rastreamento terá dificuldade de localizar um ponto específico no próximo quadro da sequência de imagens. Por outro lado, ao escolher um ponto com características únicas (inclusive de vizinhança) teremos grande chance de localizar este ponto novamente no quadro seguinte. Na prática isso significa que o ponto, ou característica, precisa ser única, ou aproximadamente única. Além disso, é necessário definir uma maneira de comparação dos pontos de uma imagem, para localizá-los em outra imagem.

O processo de rastreamento executado pelo algoritmo KLT depende de um passo de seleção de características, o qual tem como objetivo definir quais pontos da imagem original serão rastreados. Conforme apresentado no final da Seção 2.2.3, o cálculo do Fluxo Óptico pelo método clássico proposto por Lucas e Kanade é dependente da não singularidade da matriz $A^T A$, que significa dizer que ela deve possuir au-

tovalores grandes (ou maiores que um limiar). Esta condição identifica pontos na imagem que podem ser definidos como bons (ou fáceis) para rastrear (do inglês, *good features to track*, um método originalmente proposto por Shi e Tomasi [11]). Portanto, o processo de seleção destas características pode ser definido como:

1. Calcular a matriz $A^T A$ e seus autovalores mínimos λ_m, para todos os pontos da imagem I;
2. Definir λ_{max} como sendo o valor máximo de λ_m sobre toda a imagem.
3. Separar os pixels da imagem que têm λ_m maiores que uma porcentagem de λ_{max}. Esta porcentagem pode ser 10% ou 5%.
4. Destes pixels, reter o pixel de máximo local, isto é, aquele que possui o maior λ_m, de uma vizinhança de pixels 3×3.
5. Garantir uma distância mínima entre qualquer par de pixels.

Ao final do processo, os pixels restantes são denominados "bons para rastrear" e são passados para o algoritmo de rastreamento KLT.

2.2.4.3 Processo do algoritmo KLT

O processo de rastreamento piramidal do algoritmo KLT foi descrito por Bouguet [29], Minetto [1] e ilustrado por Bradski e Kaehler [3], conforme Figura 2.8. O primeiro passo consiste em executar o cálculo do Fluxo Óptico no nível mais alto da pirâmide (L_m), conforme procedimento apresentado na Seção 2.2.3. Em seguida o resultado obtido é propagado para o nível $L_m - 1$ na forma de um "chute" inicial para o deslocamento do pixel no nível $L_m - 1$. Com base neste "chute" inicial, é executado o refinamento do Fluxo Óptico no nível $L_m - 1$, que é propagado para o nível $L_m - 2$. Este procedimento é repetido até alcançar o nível zero (imagem original).

Podemos descrever formalmente o processo recursivo utilizado pelo algoritmo KLT para o cálculo do Fluxo Óptico dos níveis genéricos $L + 1$ e L. Assumindo que $\vec{v}^L = [u^L v^L]^T$ é o vetor do Fluxo Óptico calculado para nível L da pirâmide e que $\vec{c}^L = [r^L s^L]^T$ corresponde ao "chute" inicial do Fluxo Óptico do nível L, obtido da computação feita do nível $L - 1$ para o nível L. Assim, para obtermos o Fluxo Óptico do nível L é necessário encontrar o deslocamento do pixel $\vec{v}^L = [u^L v^L]^T$ que minimiza a função ϵ^L apresentada pela Equação 2.41

$$\epsilon^L(\vec{v}^L) = \epsilon^L(u^L, v^L) = \sum_{x=p_x^L-w_x}^{p_x^L+w_x} \sum_{y=p_y^L-w_y}^{p_y^L+w_y} (I^L(x,y) - J^L(x + r^L + u^L, y + s^L + v^L))^2 \qquad (2.41)$$

Cabe destacar que o tamanho da janela de integração não é alterado entre os níveis da pirâmide e o "chute" inicial irá fazer um mapeamento prévio dos pixels da imagem I na imagem J, de tal forma que o tamanho do Fluxo Óptico $\vec{v}^L = [u^L v^L]^T$ será menor e mais fácil de ser encontrado pelo método clássico proposto por Lucas e Kanade.

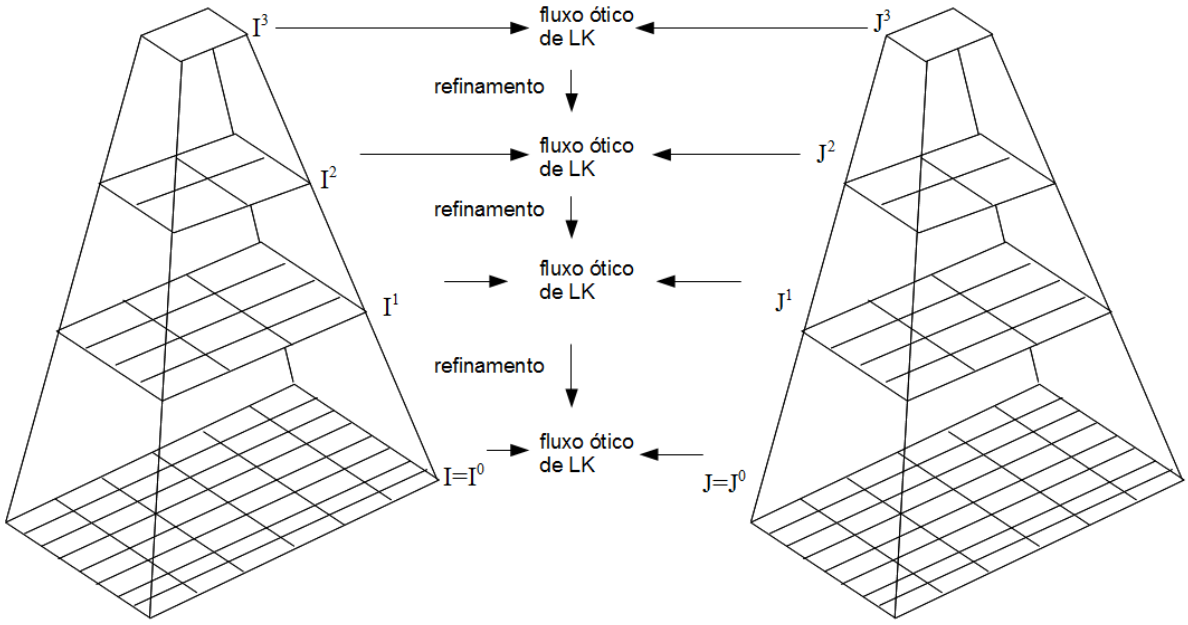


Figura 2.8: Processo piramidal utilizado no algoritmo KLT, adaptado de [3].

A propagação do "chute" inicial obtido no nível L para o nível $L-1$ é efetuada de modo que:

$$\vec{c}^{L-1} = 2(\vec{c}^{L} + \vec{v}^{L}). \tag{2.42}$$

sendo que o "chute" inicial para o nível mais alto da pirâmide é nulo ($\vec{c}^{Lm} = [0 \quad 0]^T$).

O vetor de Fluxo Óptico do próximo nível $\vec{v}^{L-1}$ é calculado da mesma maneira, minimizando a função $\epsilon^{L-1}(\vec{v}^{L-1})$ e utilizando o "chute" inicial $\vec{c}^{L-1} = [r^{L-1} s^{L-1}]^T$ que foi obtido com base nos vetores do nível L. Estes passos são repetidos até que imagem original seja encontrada. O cálculo do Fluxo Óptico final é definido como:

$$\vec{v} = \vec{c}^{0} + \vec{v}^{0}. \tag{2.43}$$

2.3 HISTOGRAMA DO FLUXO ÓPTICO ORIENTADO

Todos os métodos de cálculo do Fluxo Óptico aqui apresentados têm como resultado um conjunto de vetores que estimam a velocidade dos pixels das imagens, incluindo informações de direção e sentido do movimento. Com o objetivo de reconhecer os movimentos estimados por estes vetores de fluxo, devemos utilizar uma representação invariante a quantidade de pixels rastreados, aos ruídos presentes no fundo da imagem, às alterações de escala do objeto, bem como às mudanças de direção de um mesmo movimento. Por exemplo, ao desenvolver uma aplicação de estimação do Fluxo Óptico de ações humanas, percebe-

remos que cada ação possui um perfil de fluxo. Uma sequência de imagens com uma pessoa correndo para a direita apresentará um Fluxo Óptico com vetores indicando que os pixels estão se deslocando para a direita. Com uma pessoa pulando, teremos um fluxo de vetores na direção vertical, com o sentido variando para cima e para baixo. Ao modificarmos os objetos que executam a ação e a velocidade que ela está sendo executada, notaremos duas principais mudanças no perfil do Fluxo Óptico: a quantidade de pontos rastreados e a escala do fluxo. Entretanto, o perfil do fluxo permanecerá semelhante. Caso a alteração seja somente no sentido (pessoa correndo para a esquerda), o perfil de Fluxo Óptico será o mesmo, porém com o sentido invertido.

Para que uma aplicação de reconhecimento de padrão (que neste caso irá reconhecer tipos de movimentos, baseando-se no perfil do Fluxo Óptico) seja robusta, ela deverá implementar mecanismos que minimizem os impactos causados pela mudança do objeto, da distância do objeto em relação ao local de captura da imagem, do fundo da cena e da velocidade do movimento, parâmetros não controlados pela aplicação. Para alcançar este objetivo a maneira com que o padrão será representado deverá apresentar pouca sensibilidade à variação da escala do objeto na cena e à mudança do sentido do movimento.

Inspirados nos histogramas de características utilizados pela comunidade de reconhecimento de objetos e considerando que a característica natural de uma sequência de movimentos é o Fluxo Óptico, Chaudhry *et al.* [4] apresentaram uma metodologia para o cálculo do Histograma do Fluxo Óptico Orientado (HOOF, do inglês *Histogram of Oriented Optical Flow*). O cálculo do HOOF é executado da seguinte maneira: após a estimação do Fluxo Óptico para todos os quadros da sequência de imagens, cada vetor de fluxo é armazenado de acordo com seu ângulo primário com relação ao eixo horizontal e é ponderado de acordo com sua magnitude. Assim, todos os vetores de Fluxo Óptico, $\vec{v} = [u \quad v]^T$, com direção $\theta = \tan^{-1}(\frac{v}{u})$, no intervalo:

$$-\frac{\pi}{2} + \pi\frac{b-1}{B} \leq \theta < -\frac{\pi}{2} + \pi\frac{b}{B} \tag{2.44}$$

contribuirão com $\sqrt{u^2+v^2}$ no somatório do *bin* $b, 1 \leq b \leq B$, de um total de B *bins*. Por fim, o histograma é normalizado para somar 1. Por meio da Figura 2.9 ilustramos o procedimento para a formação do histograma com quatro *bins*, $B = 8$. A magnitude dos vetores com ângulo α serão armazenadas no *bin* 6 e a do vetor com ângulo β no *bin* 2.

Esta representação apresenta pouca variação a alterações na escala do objeto na cena em razão da normalização do histograma. Esperamos observar o mesmo histograma quando uma pessoa está acenando próximo a câmera ou quando está bem distante. Na Figura 2.10 exemplificamos esta característica da representação por HOOF. Na primeira linha da Figura apresentamos dois quadros da ação acenar com os dois braços, sendo que no primeiro a ação está sendo executada distante da câmera e no segundo a ação está próxima da câmera. Na segunda linha apresentamos as estimativas de cálculo do Fluxo Óptico utilizando o algoritmo KLT e na última linha ilustramos os histogramas dos fluxos calculados. Notamos que os histogramas gerados são semelhantes para movimentos semelhantes, mesmo quando a escala do objeto na cena é diferente. Nos casos em que a escala do objeto é um fator importante para a aplicação, esta representação não é adequada e precisa ser modificada.

O armazenamento dos vetores dos fluxos de acordo com seus ângulos primários, isto é, os menores

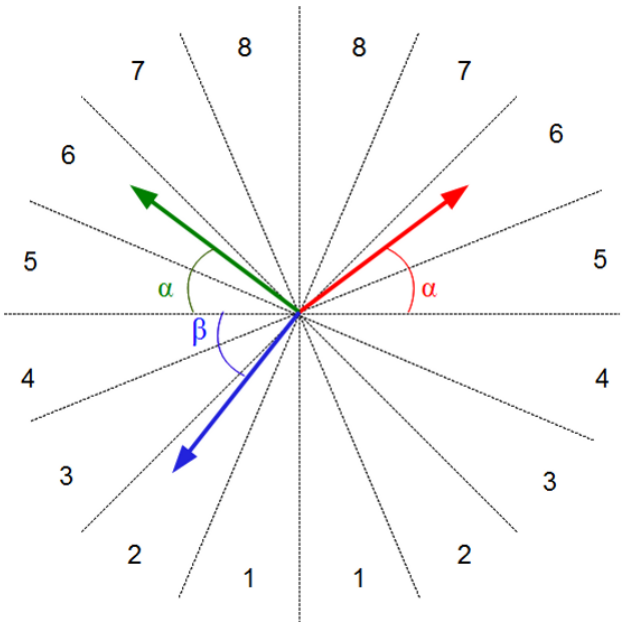


Figura 2.9: Formação do Histograma de Fluxo Óptico para oito *bins*, $B = 8$. Adaptado de [4].

ângulos entre os vetores e eixo horizontal, torna a representação pouco sensível a mudanças no sentido do movimento. Assim como no exemplo anterior, esperamos observar histogramas semelhantes quando movimentos semelhantes estão sendo executados, mesmo na ocorrência de mudança de sentido da ação. Na Figura 2.11 exemplificamos este caso. Na primeira linha ilustramos os quadros da ação correr para a direita e correr para a esquerda, na linha seguinte apresentamos as estimativas de Fluxo Óptico e na última os histogramas calculados. Observamos que mesmo quando o sentido do movimento é invertido, a representação do movimento é semelhante. Quando o sentido do movimento for importante para execução do reconhecimento, deveremos adaptar a representação por *bins*, dividindo o espaço de ângulo dos vetores em B *bins*, conforme ilustrado na Figura 2.12.

A quantidade de *bins* é um parâmetro arbitrado. Para alcançarmos bons resultados de reconhecimento, Chaudhry *et al.* [4] sugerem a utilização de pelo menos 30 *bins*.

Para cada sequência de imagens calculamos $N - 1$ estimativas de Fluxo Óptico, onde N denota a quantidade quadros da sequência de imagens, gerando $N - 1$ HOOF normalizados com a forma $h_t = [h_{t;1}, h_{t;2}, ..., h_{t;B}]^T$, obtendo uma série temporal de histogramas $\{h_t\}_{t=0}^{N-1}$. Com o objetivo de obtermos uma média temporal desta série podemos calcular a média dos histogramas $\overline{h}$ que a compõe, conforme Equação 2.45.

$$\overline{h} = \frac{1}{N} \sum_{i=0}^{N-1} h_i \tag{2.45}$$

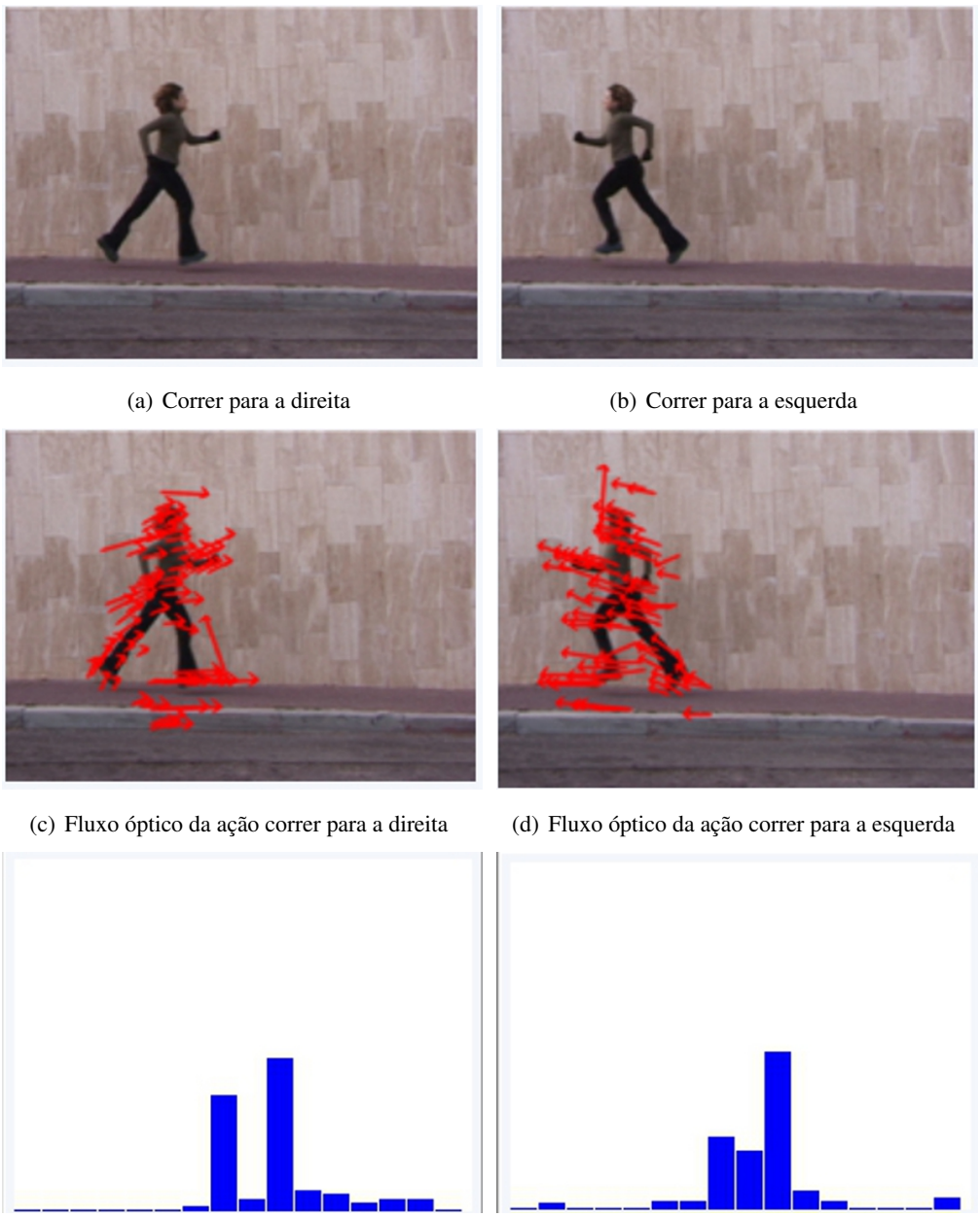
Esta média temporal pode ser utilizada para a tarefa de classificação entre as ações representadas. Sua desvantagem é que ela não representa as características da dinâmica do movimento.

(a) Acenar distante (b) Acenar perto

(c) Fluxo óptico da ação acenar distante (d) Fluxo óptico da ação acenar perto

(e) HOOF da ação acenar distante (f) HOOF da ação acenar perto

Figura 2.10: Exemplo de que o HOOF normalizado é uma representação com pouca sensibilidade à variação da escala do objeto na cena.



(a) Correr para a direita (b) Correr para a esquerda

(c) Fluxo óptico da ação correr para a direita (d) Fluxo óptico da ação correr para a esquerda

(e) HOOF da ação correr para a direita (f) HOOF da ação correr para a esquerda

Figura 2.11: Exemplo de que a representação por HOOF possui pouca variação quando da mudança do sentido do movimento.

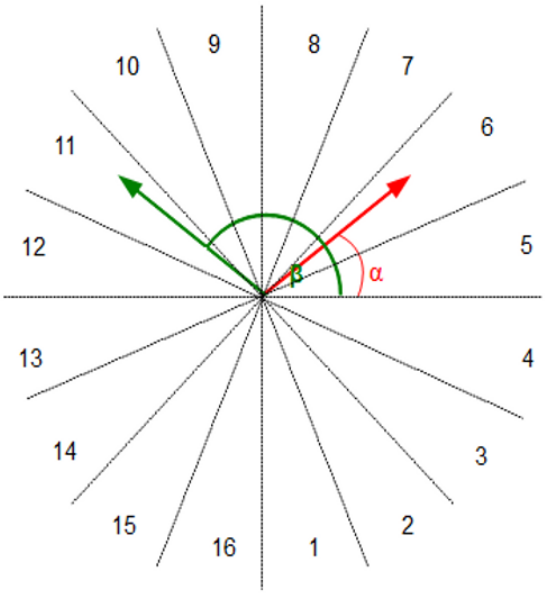


Figura 2.12: Adaptação da formação do Histograma de Fluxo Óptico para 16 *bins*, $B = 16$, com representação dependente do sentido do movimento.

2.4 CONSIDERAÇÕES FINAIS

A estimativa do Fluxo Óptico ocorrido em uma sequência de imagens é uma poderosa metodologia para obtermos informações fundamentais para o processo de avaliação de movimento. Os primeiros temas deste Capítulo abordaram os conceitos de imagem, sequência de imagens, similaridade de regiões entre imagens distintas, deslocamento óptico e Fluxo Óptico, os quais são importantes para compreensão plena dos algoritmos de cálculo e estimativa do Fluxo Óptico.

Em seguida descrevemos em detalhes os algoritmos mais utilizados para calcular o Fluxo Óptico. Também apresentamos suas deficiências e limitações. Entender o processo utilizado por cada algoritmo nos torna mais aptos na escolha de seus parâmetros, além de nos permitir escolher qual algoritmo deve ser utilizado na tarefa que desejamos implementar.

Como a quantidade de informações obtidas com o cálculo do Fluxo Óptico é bastante grande, a utilização de um método de representação capaz de estimar uma distribuição dos vetores de Fluxo Óptico irá minimizar o esforço computacional necessário para o processamento destas informações. Por este motivo, apresentamos os Histogramas de Fluxo Óptico Orientados (HOOF), descrevendo suas principais características e também os problemas e limitações associados à utilização deste tipo de representação.

No próximo Capítulo apresentamos as técnicas de Visão Computacional utilizadas para a análise do contorno do objeto no movimento. Além disso descrevemos os conceitos de relacionamento entre pixels, detecção de contorno, normalização de borda e dos descritores de contorno. Estas técnicas são importantes para o processamento e extração de informações visuais, as quais podem complementar as informações de Fluxo Óptico extraídas com os algoritmos apresentados neste Capítulo.

3 ANÁLISE DO CONTORNO DO OBJETO NO MOVIMENTO

Segundo Zhang e Lu [30] o contorno de um objeto em uma imagem é uma das características de baixo nível mais importantes para o processamento de informações visuais. Esta característica é muito importante na execução da percepção humana, que tende a perceber as cenas compondo-as em objetos individuais, os quais são melhores identificados de acordo com seus contornos. Para Costa e Cesar [6], o contorno dos objetos certamente possui um papel especial na análise de informações visuais disponíveis para a execução da percepção. Como os próprios autores exemplificam, a leitura dos caracteres nesta página só pode ser efetuada mediante a análise de seus contornos. Segundo eles, em certo sentido, os contornos podem ser consideradas como palavras da linguagem visual. A extração do contorno dos objetos de uma imagem digital e a análise deste podem prover um mecanismo robusto para a tarefa de reconhecimento de padrões.

3.1 EXTRAÇÃO DO MOVIMENTO DO OBJETO

No que tange ao reconhecimento de um movimento em uma sequência de imagens, devemos extrair o movimento do objeto para auxiliar neste reconhecimento. Na Figura 3.1 apresentamos a metodologia utilizada para a extração do movimento do objeto, a qual consiste nas etapas de:

1. conversão da imagem RBG para uma imagem em níveis de cinza;
2. cálculo da diferença absoluta entre os quadros da sequência de imagens, conforme Equação 3.1:

$$D(x, y) = |I(x, y) - J(x, y)| \tag{3.1}$$

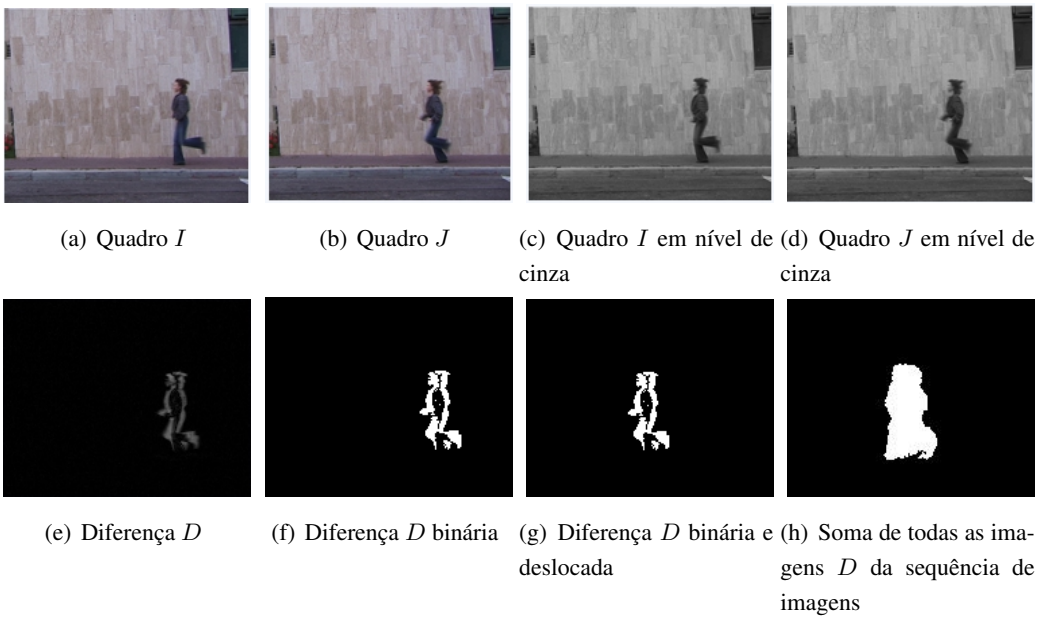
   onde I e J são quadros consecutivos na sequência de imagens, x e y correspondem às coordenadas dos pontos das imagem (ou dos quadros) e D é a diferença absoluta entre I e J;

3. binarização das imagens de diferença absoluta, calculada conforme Equação 3.2:

$$B(x, y) = \begin{cases} 1 & \text{se } I(x, y) > \text{limiar} \\ 0 & \text{caso contrário} \end{cases} \tag{3.2}$$

   onde $B(x, y)$ corresponde a diferença absoluta binária dos quadros I e J, e o limiar define um limite para a intensidade dos pixels ser considerada 0 ou 1;

4. centralização do movimento (ou objeto) detectado nas imagens binárias;
5. soma de todas as imagens de diferenças absolutas binárias e deslocadas na sequência de imagens.



(a) Quadro I (b) Quadro J (c) Quadro I em nível de cinza (d) Quadro J em nível de cinza

(e) Diferença D (f) Diferença D binária (g) Diferença D binária e deslocada (h) Soma de todas as imagens D da sequência de imagens

Figura 3.1: Metodologia utilizada para extrair o movimento do objeto.

3.2 RECUPERAÇÃO DO CONTORNO

De posse do movimento do objeto, que nada mais é do que uma imagem digital, podemos aplicar a **recuperação do contorno** nesta imagem. Nela são utilizadas técnicas de Visão Computacional para extrair os pixels de contorno (ou de borda) dos objetos na imagem, obter uma representação do contorno e apresentá-la como parâmetro de entrada a outros algoritmos de reconhecimento de padrões. Assim, a recuperação do contorno envolve definir: a metodologia que será utilizada para detectar os pixels de borda do objeto; a maneira como os contornos serão representados; a medida de similaridade entre eles; e a maneira como serão indexados [30].

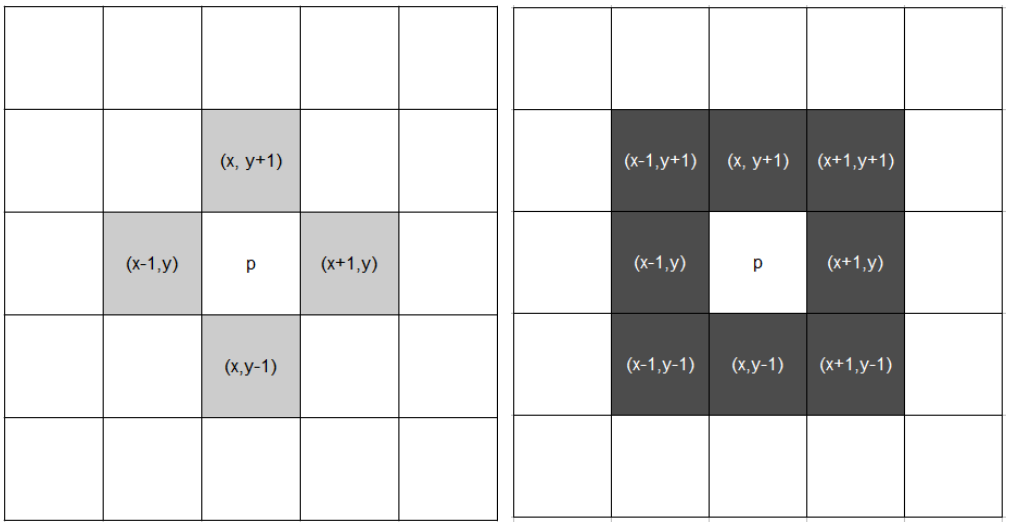
Segundo Zhange e Lu [30] a representação dos contornos, ou descritores de contorno, é a definição mais importante para a recuperação do contorno. Os métodos encontrados na literatura podem ser baseados em regiões ou baseados em borda. As técnicas baseadas em regiões consideram todos os pixels do objeto, que são contabilizados para obter uma representação. É comum utilizar descritores de momento para representar estes contornos, os quais interpretam os níveis de cinza normalizados de uma imagem como uma função de densidade de probabilidade de uma variável aleatória bidimensional. As técnicas baseadas em borda são mais populares e exploram somente os pixels de borda do objeto. Estes métodos baseados em borda são comumente divididos em descritores globais, assinaturas de contorno e descritores espectrais. Os descritores globais são fáceis de computar e utilizam informações como área, circularidade e orientação. As assinaturas de contorno podem utilizar coordenadas complexas, curvaturas, distâncias e informações angulares. Os descritores espectrais tratam-se de procedimentos executados com as assinaturas do contorno, usando transformadas espectrais, tais como a Transformada de Fourier e a Transformada Wavelet.

Nas próximas Seções apresentamos os conceitos de relacionamento entre pixels, o processo de detecção dos pixels de borda e as metodologias de representação da borda extraída.

3.3 RELACIONAMENTOS ENTRE PIXELS

Em uma imagem digital, que neste contexto pode ser expressa por $f(x, y)$, existem algumas relações básicas entre pixels, cujos conceitos possuem um papel importante na compreensão de diversas técnicas de processamento de imagens e Visão Computacional. Entre estas relações podemos citar a vizinhança, a adjacência, os caminhos, a conectividade, as regiões e as fronteiras.

De acordo com Gonzalez e Woods [25], um pixel p na coordenada (x, y) possui quatro vizinhos horizontais e verticais que possuem as coordenadas $(x+1, y)$, $(x-1, y)$, $(x, y+1)$ e $(x, y-1)$. Este conjunto de pixels é denominado *vizinhança-4* de p. Além disso, p também possui outros 4 vizinhos diagonais, com coordenadas $(x+1, y+1)$, $(x+1, y-1)$, $(x-1, y+1)$ e $(x-1, y-1)$, que juntamente com os pixels do conjunto *vizinhança-4* compõe o conjunto *vizinhança-8*. As Figuras 3.2(a) e 3.2(b) ilustram os conjuntos *vizinhança-4* e *vizinhança-8*, respectivamente.



(a) vizinhança-4 (b) vizinhança-8

Figura 3.2: Conjunto de pixels que compõe as vizinhanças *4* e *8* do *pixel* p.(a)*Pixels* pertencentes a *vizinhança-4*, marcados com a cor cinza. (b)*Pixels* pertencentes a *vizinhança-8*, marcados com a cor cinza escuro.

O conceito de adjacência depende destes conceitos de vizinhança. No contexto deste trabalho, utilizaremos dois tipos de adjacência: a *adjacência-4* e a *adjacência-8*. Considerando que V é o conjunto de valores de intensidade utilizados para definir a adjacência, para definirmos os pixels adjacentes com valor igual a 1 em uma imagem binária, definimos $V = 1$. Dizemos que dois pixels p e q, com valores pertencentes a V, são *adjacentes-4* se q estiver no conjunto *vizinhança-4* de p. De maneira análoga, dizemos que dois pixels p e q, com valores pertencentes a V, são *adjacentes-8* se q estiver no conjunto *vizinhança-8* de p.

Conforme apresentado por Gonzalez e Woods [25], um caminho do pixel p com coordenadas (x, y) ao pixel q com coordenadas (s, t) é uma sequência de pixels distintos com coordenadas:

$$(x_0, y_0), (x_1, y_1), ..., (x_n, y_n) \tag{3.3}$$

onde $(x_0, y_0) = (x, y)$, $(x_n, y_n) = (s, t)$ e os pixels (x_i, y_i) e (x_{i-1}, y_{i-1}) são adjacentes para $1 \leq i \leq n$, sendo n o comprimento do caminho. Se $(x_0, y_0) = (x_n, y_n)$ dizemos que o caminho é fechado. Portanto, podemos definir caminhos-4 ou caminhos-8 a depender do tipo de adjacência que for especificada.

Considerando S um subconjunto de pixels, dizemos que p e q são conexos em S se existir um caminho entre eles, composto exclusivamente por pixels em S. Tomando qualquer pixel p em S, o conjunto de pixels que são conectados a ele em S é denominado componente conexo de S. Existindo somente um componente conexo, chamamos o conjunto S de conjunto conexo.

Tomando um subconjunto de pixels R de uma imagem, caso R seja um conjunto conexo chamamos R de uma região da imagem. Duas regiões são consideradas adjacentes se sua união formar um conjunto conexo. Caso não forem adjacentes, serão consideradas regiões disjuntas. Destacamos que sempre será necessário associar a uma região o tipo de conectividade considerado. Supondo que uma imagem possua K ($k = 1, 2, 3, ..., K$) regiões disjuntas, dizemos que a união de todas as regiões K (R_u) é a frente da imagem e o complemento desta união (R_u^c) é o fundo da imagem.

A fronteira, borda ou contorno de uma região é o conjunto de pixels desta região que tem pelo menos um vizinho no fundo da imagem, ou seja, o conjunto de pontos de R adjacentes aos pontos do complemento de R. Novamente, o tipo de conectividade para definir a adjacência deve ser especificado. Alguns autores, a depender do contexto que estão trabalhando, costumam distinguir entre os termos borda e fronteira. Conforme apresentado por Gonzalez e Woods [25], a fronteira de uma região finita forma um caminho fechado, sendo assim considerado um conceito global. As bordas são formadas por pixels com valores cujas derivadas excedem um limiar pré-definido. Desta forma a borda é um conceito "local" baseado em medidas de descontinuidade. Em imagens binárias a borda e as fronteiras são correspondentes. Como neste trabalho utilizaremos apenas imagens binárias, para efetuar a análise do contorno dos objetos, consideraremos os termos borda e fronteira com o mesmo sentido.

3.4 DETECÇÃO DE CONTORNO

Também conhecido como Detecção de Bordas, o processo de Detecção de Contorno é uma técnica aplicada em imagens digitais com o objetivo de extrair a borda das imagens. É considerada uma técnica de pré-processamento de imagens digitais, visto que seu objetivo é extrair informações genéricas acerca do contorno dos objetos ou padrões presentes na imagem. Com o contorno de um determinado padrão podemos mapear suas características e utilizá-las num processo posterior que irá classificá-lo. Quanto mais precisa for esta detecção, maior a probabilidade de alcançarmos resultados mais acurados na execução do processo de classificação. Outro fator positivo na utilização da informação de borda para o reconhecimento de padrões é a redução do esforço computacional. Em geral, a quantidade de pixels de contorno é bem inferior ao total de pixels da imagem. Assim, quando utilizamos apenas os pixels do contorno do objeto estamos reduzindo o esforço computacional necessário para a classificação do padrão [5].

Na literatura podemos encontrar diversos algoritmos que implementam esta técnica [5] [31] [32] [33]. Os principais são: *Square Tracing Algorithm*, *Moore-Neighbor Tracing*, *Radial Sweeep* e *Theo Pavlidis' Algorithm*. Neste trabalho utilizamos o algoritmo *Moore-Neighbor Tracing*, descrito na próxima Seção,

que está implementado e disponível para utilização nas bibliotecas do Matlab. Cabe salientar que todos os algoritmos possuem suas próprias deficiências e podem apresentar falhas no rastreamento de contornos, a depender do tipo de padrão ou do tipo de conectividade escolhida.

3.4.1 *Moore-Neighbor Tracing*

Iniciamos esta Seção com a definição da **Vizinhança de Moore**, conceito fundamental para a compreensão do processo do algoritmo *Moore-Neighbor Tracing*. A Vizinhança de Moore de um pixel P é o conjunto de oito pixels pertencentes a vizinhança-8 do pixels P, ou seja, são aqueles que compartilham um vértice ou uma aresta com o pixel P. Eles são denominados pixels $P1$, $P2$, $P3$, $P4$, $P5$, $P6$, $P7$, $P8$, conforme ilustrado na Figura 3.3.

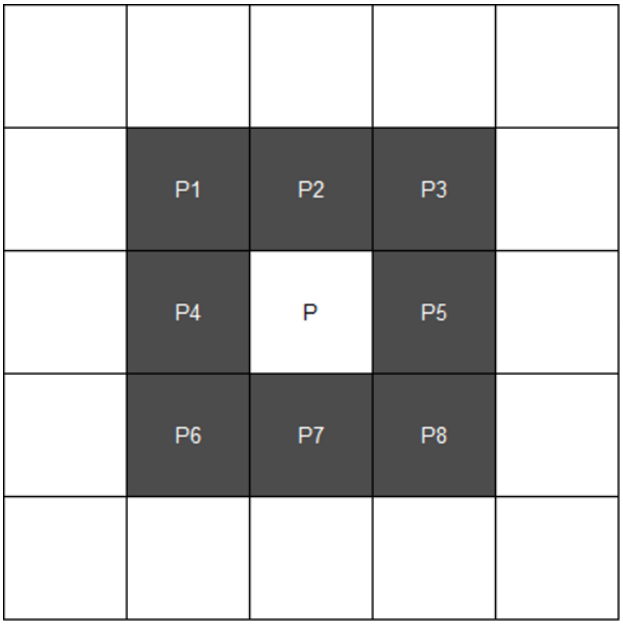


Figura 3.3: Vizinhança de Moore. Adaptado de [5].

Dado um padrão em uma imagem digital, que é composto por um grupo de pixels pretos, numa imagem com fundo de pixels brancos, o primeiro passo do algoritmo de Moore é localizar um pixel preto e declará-lo como sendo o **pixel de partida** P. Este processo de localização do primeiro pixel preto pode ser feito de diversas maneiras. Uma delas é varrer a imagem do canto inferior esquerdo para o canto superior direito, examinando cada coluna de cada linha até encontrar o pixel de partida. Os passos seguintes consistem em localizar os pixels pertencentes ao contorno do padrão examinando os pixels da Vizinhança de Moore dos pixels pretos que forem sendo localizados.

Na Figura 3.4 apresentamos o processo executado por este algoritmo. O padrão cujo contorno deseja ser localizado está apresentado na Figura 3.4 (a). Conforme descrito anteriormente, o primeiro passo é a localização do pixel de partida P, ilustrada nas Figuras 3.4 (b) e (c). Para a compreensão do algoritmo, consideramos a existência de um ponteiro indicando o pixel da imagem digital que está sendo examinado. Após a localização do pixel P, retornamos a posição deste ponteiro para o pixel imediatamente anterior à localização do pixel de partida, conforme Figura 3.4 (d). A partir deste ponto, examinamos os pixels

pertencentes a Vizinhança de Moore que estão ao redor do pixel P até encontrar um outro pixel preto, conforme Figura 3.4 (e). Neste exemplo estamos examinando a Vizinhança de Moore no sentido horário e, por consequência, o contorno também será obtido no sentido horário. O pixel de partida P e o pixel corrente, denominado pixel C, devem ser marcados como pertencentes a borda do padrão. Os passos de retorno do ponteiro, exame da Vizinhança de Moore e localização de um novo pixel C são executados novamente até que $C = P$, ou seja, até que o pixel de partida seja visitado pela segunda vez, conforme ilustrado pelas Figuras 3.4 (f) a (l).

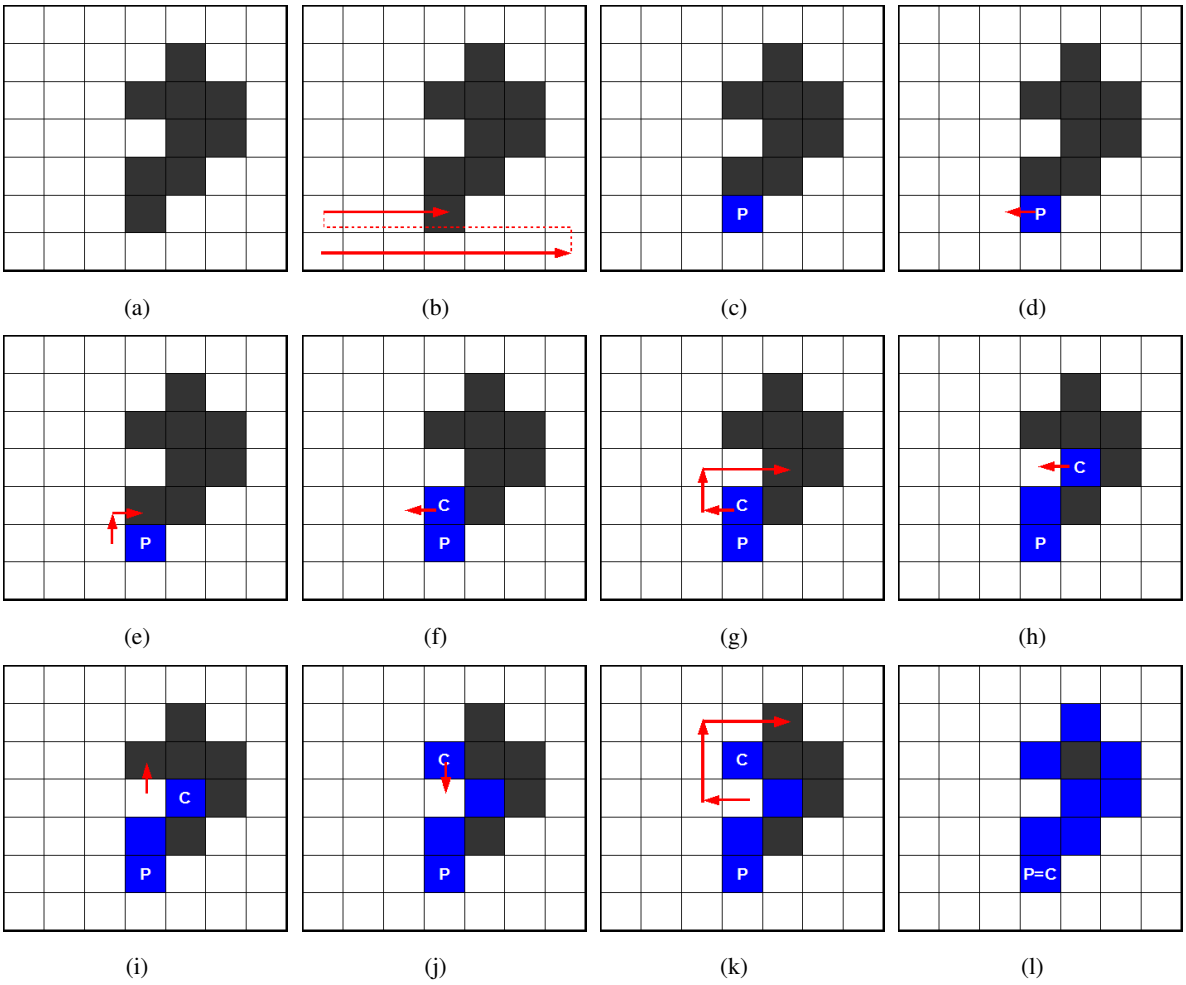


Figura 3.4: Processo de Rastreamento de Contorno segundo o algoritmo *Moore-Neighbor Tracing*. Adaptado de [5].

O algoritmo de rastreamento de contorno *Moore-Neighbor Tracing* pode ser formalmente descrito conforme o Algoritmo 1. Considere $M(a)$ a Vizinhança de Moore do pixel a, c como o pixel de borda corrente e v denota o pixel atual em consideração, estando v em $M(c)$.

Algoritmo 1: Algoritmo de rastreamento de contorno *Moore-Neighbor Tracing*. Adaptado de [5].

Entrada: Uma imagem digital, I, a qual contém um componente conectado C de pixels pretos.

Saída: Uma sequência

$B = (b_0 = (x(0), y(0)), b_1 = (x(1), y(1)), ..., b_{L-1} = (x(L-1), y(L-1)))$,

$t = 0, 1, ..., L-1$, de pixels de borda, isto é, o contorno contendo L pixels.

Defina B vazio

De baixo para cima e da esquerda para a direita, examine os pixels de I até encontrar um pixel preto p (pixel de partida), que pertence a C

Insira as coordenadas $(x(p), y(p))$ do pixel p em B

Defina p como o pixel corrente c, ou seja $c = p$

Retorne para o pixel anterior, por onde o pixel p foi encontrado

Defina o próximo pixel de $M(c)$ no sentido horário como v

enquanto $v \neq p$ **faça**

- **se** v *é preto* **então**
  - Insira as coordenadas $(x(v), y(v))$ do pixel v em B
  - Defina $c = v$
  - Retorne para o pixel anterior, por onde o pixel c foi encontrado
- **senão**
  - Avance o pixel atual v para o próximo pixel de $M(c)$ no sentido horário

A saída do algoritmo são os pixels de borda de C e k indica a quantidade pixels de contorno que foram encontrados para C. Segundo Ghunein [5] a principal deficiência do algoritmo *Moore-Neighbor Tracing* é o seu critério de parada (quando seu pixel de partida for visitado pela segunda vez). Este algoritmo pode falhar em grande quantidade de padrões, em razão da escolha deste critério. Na Figura 3.5 apresentamos um exemplo onde a extração do contorno de um padrão pode falhar, caso o critério de parada tradicional seja escolhido.

Ghunein [5] apresenta dois critérios de parada alternativos:

1. Parar a execução quando o pixel de partida for visitado n vezes, sendo $n \geq 2$;
2. Parar após entrar no pixel de partida uma segunda vez, mas da mesma maneira que entrou inicialmente. Este critério foi proposto por *Jacob Eliosoff* e é conhecido como *Jacob's stopping criterion*.

O critério de parada de *Jacob* produz resultados melhores do que aqueles alcançados com o critério tradicional. Quando o algoritmo visitar o pixel de partida pela segunda vez, da mesma maneira que inicialmente, ele já terá traçado o contorno completo. Caso não seja interrompido, irá traçar o mesmo contorno novamente. Na Figura 3.6 apresentamos o resultado do rastreamento do mesmo objeto da Figura 3.5 utilizando o critério de parada de *Jacob*.

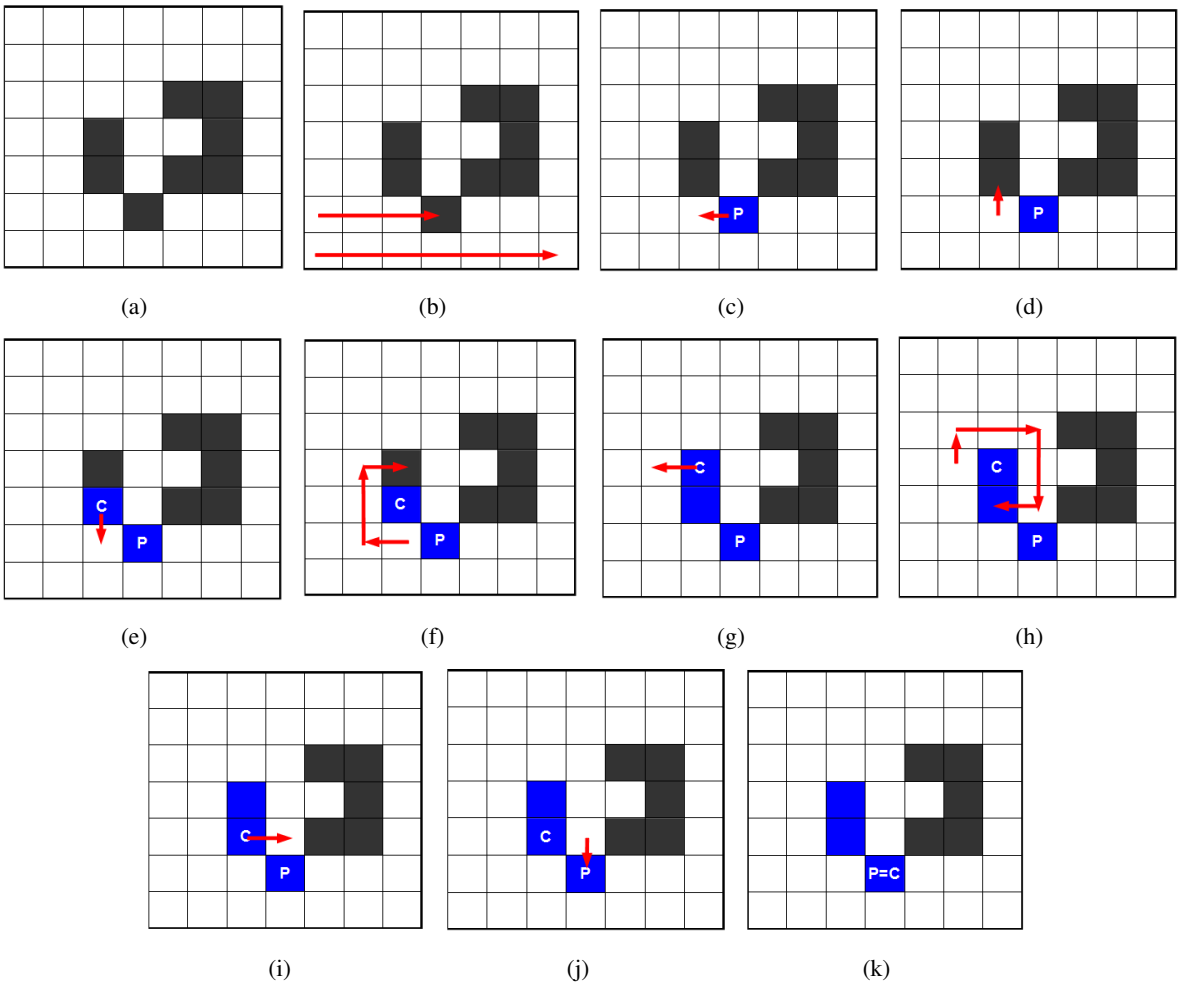


Figura 3.5: Exemplo de falha no processo de Rastreamento de Contorno segundo o algoritmo *Moore-Neighbor Tracing*. Adaptado de [5].

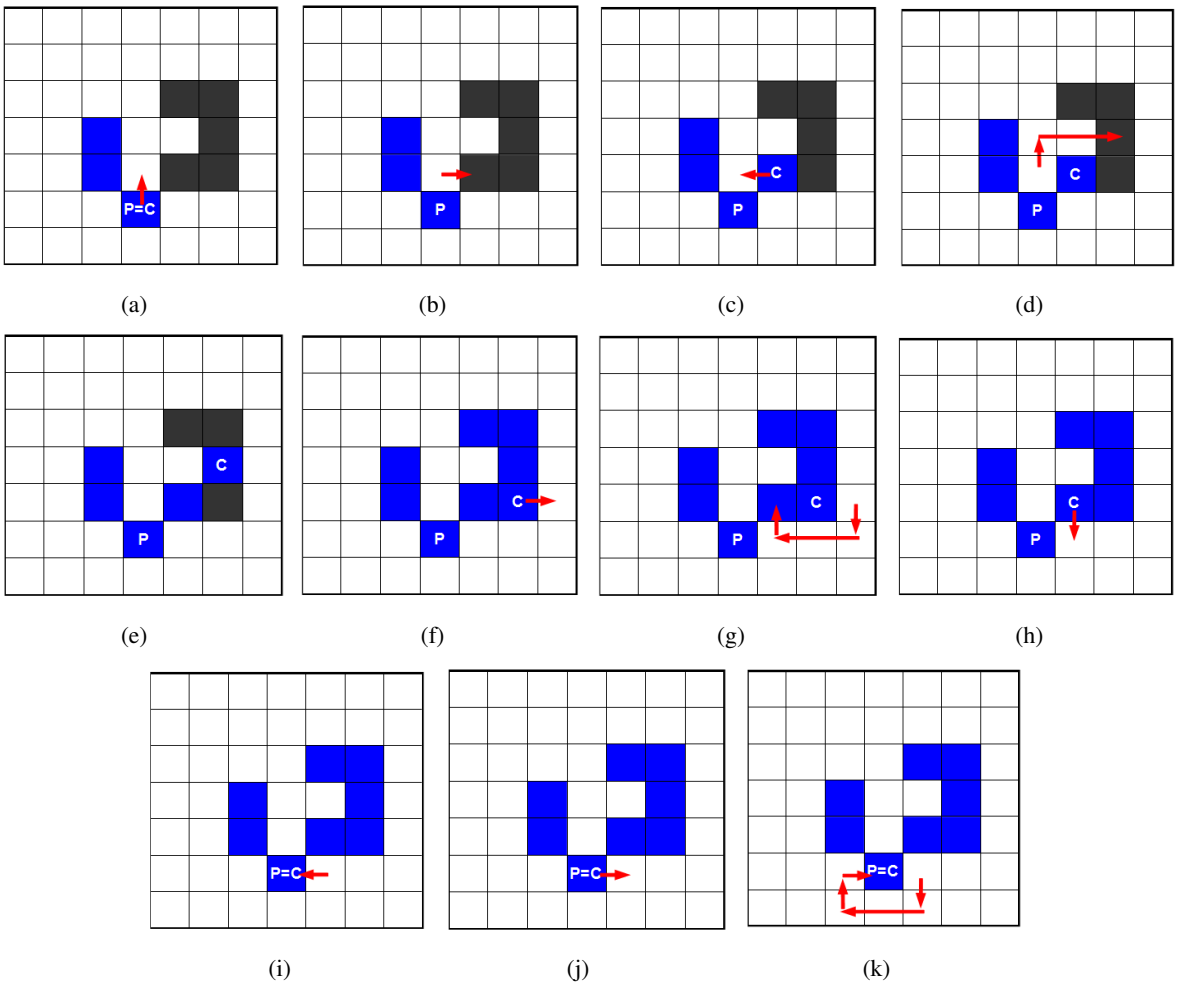


Figura 3.6: Exemplo do processo de Rastreamento de Contorno segundo o algoritmo *Moore-Neighbor Tracing*, utilizando o critério de parada de *Jacob*. Adaptado de [5].

3.5 ASSINATURAS DE CONTORNO

Com o objetivo de gerar uma representação global do contorno podemos utilizar as Assinaturas de Contorno, que são funções unidimensionais que representam uma área ou borda bidimensional. Para Gonzalez e Woods [25] as funções que geram as assinaturas buscam reduzir a representação da fronteira para uma função 1-D, que seja mais fácil de descrever do que a fronteira 2-D original. Zhang e Lu [30] apresentam quatro técnicas de Assinatura de Contorno: assinatura de distância do centroide; assinatura de coordenadas complexas; assinaturas de curvatura; e assinaturas de função angular acumulada. Os autores apresentaram resultados comparando experimentos de recuperação de contorno utilizando estas quatro técnicas de assinatura. A assinatura de distância de centroide, invariante à translação, foi a que apresentou resultados melhores e mais robustos, capturando características locais e globais do contorno.

Assumindo que as coordenadas dos pixels de borda $(x(t), y(t)), t = 0, 1, ..., L-1$, foram extraídas com a execução do algoritmo de rastreamento de contorno, podemos calcular a assinatura baseada na distância do centroide conforme apresentado na Equação 3.4, que calcula a distância euclidiana de cada pixel de contorno ao centroide do objeto.

$$r(t) = ([x(t) - x_c]^2 + [y(t) - y_c]^2)^{\frac{1}{2}} \tag{3.4}$$

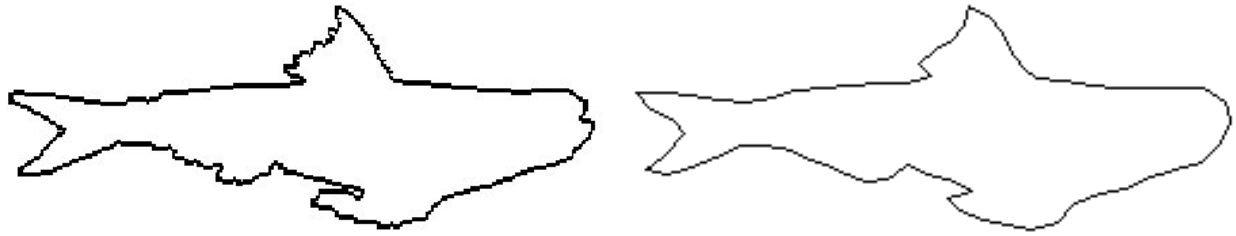
onde (x_c, y_c) é o centroide do contorno, que é calculado com base na média das coordenadas de borda, conforme Equação 3.5.

$$x_c = \frac{1}{L}\sum_{t=0}^{L-1} x(t), y_c = \frac{1}{L}\sum_{t=0}^{L-1} y(t) \tag{3.5}$$

Esta metodologia é invariante à translação, porém é dependente da rotação e da escala do objeto que está representando. Uma forma de minimizar uma destas dependências é normalizar em relação a rotação, encontrando uma maneira de selecionar o mesmo ponto de partida para gerar a assinatura, independente da orientação do contorno. Como ponto de partida podemos escolher: o ponto mais distante do centroide; o ponto mais próximo do centroide; o ponto mais próximo de um dos cantos da imagem; ou um ponto sobre o auto-eixo da imagem que estiver mais próximo do centroide. O problema da escala pode ser contornado buscando um método que ajuste todas as funções que mapeiem suas as entradas em saídas no intervalo $[0, 1]$, removendo do método a dependência ao tamanho e preservando a forma fundamental da função [25].

3.6 NORMALIZAÇÃO DA BORDA

O algoritmo de rastreamento de contorno *Moore-Neighbor Tracing* define as coordenadas dos pixels de borda das imagens. O resultado deste algoritmo produz uma borda B, cuja dimensão, ou quantidade de pontos de borda, é dependente do tamanho e do contorno dos objetos da imagem. Para permitir a comparação entre os diversos padrões de bordas gerados, ou de suas respectivas assinaturas, é importante que os



(a) (b)

Figura 3.7: Exemplo de normalização da borda. (a) contorno original. (b) contorno com borda normalizada [30].

modelos de representação utilizados possuam a mesma quantidade de pontos. Para atender a este requisito, devemos executar a normalização das bordas dos objetos. O processo de normalização de borda, além de igualar a quantidade de pontos dos padrões, também irá permitir a suavização do contorno. A suavização da borda serve para a eliminação de ruídos e de pequenos detalhes presentes na borda. Segundo Zhang e Lu [30], a variação na quantidade de pontos irá permitir modificar o nível de detalhes da representação do contorno. Com uma grande quantidade de pontos, mais detalhes serão representados e consequentemente os resultados de comparação entre os padrões serão mais acurados. Por outro lado, com poucos pontos reduzimos o nível de detalhes utilizados na comparação, mas melhoramos a eficiência computacional do processo executado pelo algoritmo de comparação. Os autores apresentam três métodos de normalização de borda: normalização por igualdade de quantidade de pontos; normalização por igualdade angular; e normalização por igualdade de comprimento de arco.

Considerando K a quantidade de pontos que a borda terá, a normalização por igualdade de pontos seleciona os K pontos igualmente distribuídos pela borda, avaliando o espaço entre pontos consecutivos, que é dado por $\frac{L}{K}$, onde L é a quantidade de pontos da borda. A normalização por igualdade angular executa um processo semelhante, porém os pontos candidatos a pertencerem a borda normalizada são avaliados de acordo o ângulo entre pontos consecutivos. Os pontos selecionados possuem distância angular $\theta = \frac{2\pi}{K}$. Já a normalização por igualidade de comprimento de arco avalia os pontos candidatos de acordo com o comprimento do arco que os separa. Para que eles sejam selecionados, o comprimento do arco entre eles deve ser igual a $\frac{P}{K}$, onde P é o perímetro da borda do contorno.

Zhang e Lu [30] apresentam um exemplo de normalização de um contorno que representa um tubarão, utilizando o método de normalização por igualdade de comprimento de arco, ilustrado na Figura 3.7. Segundo eles este método é o que apresenta o melhor efeito de distribuição igualitária dos pontos pela borda do contorno. Podemos notar que o efeito direto da normalização é a suavização do contorno, com a eliminação dos ruídos e de pequenos detalhes, sem afetar a robustez da representação do objeto pelo seu contorno, preservando, neste caso, as características principais do objeto, ou seja, as extremidades do corpo do tubarão (cauda, barbatanas e formato do corpo).

3.7 DESCRITORES DE FOURIER

Segundo Costa e Cesar [6] o método de representação de contorno baseado nos Descritores de Fourier é um dos mais populares em aplicações de visão e de reconhecimento de padrões. Os Descritores de Fourier podem ser obtidos de diversas maneiras e são considerados uma classe de métodos. Costa e Cesar [6] afirmam que a ideia básica destes métodos consiste em representar o contorno de interesse em termos de um sinal uni ou bidimensional e tomar a Transformada de Fourier deste sinal, calculando os Descritores de Fourier desta representação. Assim, passam a representar o contorno do objeto no domínio da frequência. Os descritores de frequência baixa contém informações acerca das características gerais do contorno, enquanto os de frequência alta contém informações sobre os detalhes do contorno. Zhang e Lu [30] apresentam um método que utiliza as assinaturas de contorno descritas na Seção 3.5. Assumindo que as assinaturas de contorno $s(t)$, onde $t = 0, 1, ..., L$, estão normalizadas com N pontos, conforme procedimento descrito na Seção 3.6, a transformada discreta de Fourier de $s(t)$ é obtida conforme Equação 3.6.

$$u(n) = \frac{1}{N} \sum_{t=0}^{N-1} s(t) e^{\frac{-j2\pi nt}{N}}, n = 0, 1, ..., N-1 \tag{3.6}$$

Os coeficientes complexos $u(n)$, com $n = 0, 1, ..., N-1$, são os Descritores de Fourier da fronteira, ou borda do contorno, denotados por FD_n, com $n = 0, 1, ..., N-1$. A transformada inversa de Fourier desses coeficientes, obtida através da Equação 3.7, reconstrói $s(t)$.

$$s(t) = \frac{1}{N} \sum_{n=0}^{N-1} u(n) e^{\frac{j2\pi nt}{N}}, t = 0, 1, ..., N-1 \tag{3.7}$$

Gonzalez e Woods [25] apresentam um método que trata cada par de coordenadas como um número complexo, conforme Equação 3.8.

$$s(t) = x(t) + jy(t) \tag{3.8}$$

Onde $t = 0, 1, ..., L$. Desta forma, o problema 2-D foi reduzido para um problema 1-D, sendo o eixo x tratado como eixo real e o eixo y como o eixo imaginário de uma sequência de números complexos. Os autores afirmam que embora a interpretação da fronteira tenha sido reformulada, sua natureza em si não foi alterada. Neste caso, a transformada inversa de Fourier reconstrói a borda da imagem.

De acordo com Gonzalez e Woods [25], poucos descritores de Fourier são suficientes para capturar as propriedades fundamentais da fronteira, e podem ser utilizados como base para permitir a comparação entre diferentes formatos de fronteiras. Quanto menor a quantidade de descritores de Fourier, mais detalhes serão perdidos.

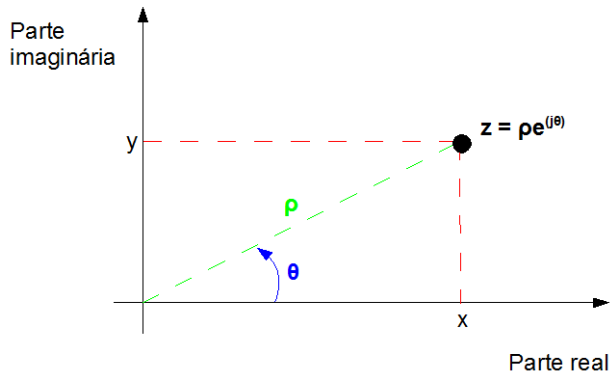


Figura 3.8: Número complexo na forma polar, plotado em um plano complexo. Adaptado de [6].

3.7.1 Descritores de Fourier Invariantes a Escala, Rotação e Translação

A recuperação de contornos busca principalmente identificar as características principais do contorno e através delas compará-las com outros contornos disponíveis. Em geral, informações de posição, tamanho e rotação não são tão importantes. Segundo Zhang e Lu [30], para obtermos um modelo de contorno e torná-lo comparável a um conjunto de dados de contorno, a representação utilizada deve ser invariante a translação, rotação e escala. No domínio da frequência é relativamente fácil obtermos representações que atendam a estes requisitos.

Considerando que os métodos de obtenção das assinaturas das bordas dos contornos, descritos na Seção 3.5, são invariantes à translação, o método de cálculo dos descritores de Fourier baseados nas assinaturas de bordas também são invariantes à translação.

Para obtermos descritores de Fourier invariantes a rotação devemos ignorar a informação do argumento do número complexo e considerar somente a informação de módulo. Lembrando que, considerando z um número complexo, e sendo $z = x+jy$, onde x é parte real do número complexo e y sua parte imaginária. A forma polar de número complexo, ilustrada na Figura 3.8, é dada por $z = \rho e^{j\theta}$, onde $|z| = \rho$ é a magnitude (ou módulo) de z e $\phi(z) = \theta$ é o seu argumento. Mediante técnicas da trigonometria básica podemos notar que a parte real do número complexo é dada por $\rho\cos(\theta)$ e a parte imaginária por $\rho\sin(\theta)$, sendo $z = \rho[\cos(\theta) + j\sin(\theta)] = x + jy$, $\rho = \sqrt{x^2 + y^2}$ e $\theta = \arctan(\frac{y}{x})$.

A obtenção dos descritores de Fourier invariantes a escala dependerá do método de assinatura de borda utilizado. Para a assinatura por distância do centroide, considerando que a Equação 3.4 apresenta apenas resultados reais, somente $N/2$ frequências diferentes são obtidas com a transformada de Fourier. Assim, apenas metade dos descritores são necessários para indexar o contorno. A normalização escalar é obtida dividindo os valores de magnitude dos descritores de Fourier, $|u(n)|$, pelo componente primeiro coeficiente do descritor (componente DC), conforme Equação 3.9.

$$f = [\frac{|u(1)|}{|u(0)|}, \frac{|u(2)|}{|u(0)|}, ..., \frac{|u(N/2)|}{|u(0)|}] \tag{3.9}$$

Com os descritores normalizados e invariantes a escala, rotação e translação podemos verificar a similaridade entre um modelo de contorno, indexado pelos descritores de Fourier $f_m = [f_m^1, f_m^2, ..., f_m^{Nc}]$, e

um objeto qualquer indexado pelos descritores $f_d = [f_d^1, f_d^2, ..., f_d^{Nc}]$, através da distância euclidiana entre eles, apresentada na Equação 3.10. Considere Nc o número de componentes do descritor necessários para indexar o contorno.

$$d = (\sum_{i=0}^{Nc} |f_m^i - f_d^i|^2)^{\frac{1}{2}} \tag{3.10}$$

3.8 CONSIDERAÇÕES FINAIS

O contorno de um objeto em uma imagem é uma característica valiosa para a execução do processamento de dados visuais, utilizados na Visão Computacional. Ele se torna especialmente importante quando nos referimos ao desenvolvimento de aplicações de percepção visual. As tarefas de **detecção do contorno** dos objetos, de **geração de representações** que sumarizem o contorno detectado e a **análise** de suas características através de algoritmos de reconhecimento de padrão, permitem a extração de informações relevantes à análise e ao reconhecimento de movimentos. Neste Capítulo buscamos descrever as técnicas utilizadas para implementar as duas primeiras tarefas citadas.

Iniciamos o Capítulo descrevendo o processo de extração do movimento do objeto, através do qual sumarizamos todo o movimento ocorrido na sequência de imagens em uma única imagem, que contém um objeto representante do movimento. Ao aplicar o algoritmo de **detecção de contorno** *Moore-Neighbor Tracing* obtemos as coordenadas dos pontos de borda do objeto representante do movimento. Para compreensão deste algoritmo foi necessário apresentar os conceitos referentes aos relacionamentos entre os pixels de uma imagem.

De posse das coordenadas dos pontos de borda, aplicamos uma função que normaliza a quantidade de pontos encontrados e outra função que mapeia cada coordenada normalizada em uma **representação** numérica, gerando um padrão para cada movimento.

A tarefa de **análise** das representações geradas é executada através de algoritmos de reconhecimento de padrão, comumente utilizados em aplicações de Inteligência Computacional. No próximo Capítulo apresentaremos as Redes Neurais Artificiais que são metodologias capazes de gerar uma modelagem matemática para os padrões de entrada apresentados à rede. Com esta modelagem as redes podem classificar ou agrupar os padrões de entrada, avaliando suas características e comparando-as com aquelas sumarizadas em seu modelo matemático.

4 REDES NEURAIS ARTIFICIAIS

Conforme já discutimos, temos vivenciado o desenvolvimento das engenharias e da computação, o que nos torna capazes de produzir em alta velocidade grande quantidade de aplicações e de dados, os quais podem ser transmitidos rapidamente a qualquer parte do mundo através da rede mundial de computadores. A evolução de tais tecnologias tem demandado o desenvolvimento de máquinas mais autônomas, capazes de emular um comportamento dito inteligente, afim de automatizar processos, agilizando e contribuindo para a tomada de decisão por parte dos homens, reduzindo a necessidade de recursos humanos para a execução de tarefas tidas como cotidianas, que exijam raciocínio, aprendizagem e planejamento. Na computação e na engenharia existe uma área de pesquisa que busca desenvolver algoritmos para a solução de problemas cuja solução envolve aspectos da inteligência humana, criando modelos capazes de emular um comportamento inteligente, exigindo a compreensão do que é um comportamento computacionalmente inteligente, para a construção de ferramentas que sejam capazes de o executarem através de métodos que apliquem raciocínio, aprendizagem e planejamento. Esta área é a Inteligência Artificial (IA) e ela utiliza conceitos da Ciência da Computação, da Matemática, da Biologia, da Neurociência, da Cibernética, da Linguística, da Economia, da Filosofia, para implementação dos modelos computacionais capazes de simular um comportamento inteligente. Cabe destacar que não existe um consenso acerca da definição dos procedimentos computacionais que nós (humanos) queremos chamar de inteligentes.

Dentre as diversas áreas de pesquisa da IA encontramos a IA conexionista que utiliza uma abordagem neural inspirada na biologia para executar a aprendizagem. Esta linha de pesquisa busca simular a inteligência humana através de modelos computacionais que atuam de maneira semelhante à estrutura e ao funcionamento do cérebro, em especial dos neurônios e de suas conexões. O comportamento inteligente é obtido em sistemas de componentes simples, interativos, que passam por um processo de aprendizagem, ou adaptação, pelo qual as conexões entre os componentes são ajustadas para modelar um domínio de problema específico. Suas principais ferramentas são as Redes Neurais Artificiais (RNA), modelos matemáticos que utilizam a ideia de neurônios como unidades de processamento interconectadas por sinapses ponderadas, cujos conceitos e definições serão apresentados no decorrer deste Capítulo. As RNA são comumente utilizadas para executar tarefas de classificação, reconhecimento de padrões, evocação de memória, predição, otimização, e filtragem de ruído. Apresentamos nas Seções seguintes os conceitos e definições das RNAs e em seguida as etapas comumente utilizadas para o desenvolvimento de aplicações utilizando as RNA.

4.1 CONCEITOS E DEFINIÇÕES DAS REDES NEURAIS ARTIFICIAIS

As Redes Neurais Artificiais (RNA), também referenciadas simplesmente como Redes Neurais, são modelos matemáticos inspirados nas redes neurais biológicas, comumente utilizadas para executar tarefas de processamento de dados para obtenção e extração de informações que poderão servir como entrada de algum processo ou auxiliar na tomada de decisão humana. Luger [34] relaciona as tarefas computacionais

para as quais a utilização de RNA é adequada: classificação, reconhecimento de padrões, evocação de memória, predição, otimização e filtragem de ruído.

Segundo Haykin [8], uma rede neural é um processador maciçamente paralelamente distribuído, composto de unidades de processamento simples, os quais possuem propensão natural para o armazenamento de conhecimento experimental e torná-lo disponível para uso. Para Fausett [35], uma RNA é um sistema de processamento de informação que possui algumas características de execução semelhante com as redes neurais biológicas. Suas características fundamentais que fazem com que se assemelhe ao cérebro são:

1. O conhecimento é adquirido pela rede, partindo do ambiente através de um processo de aprendizagem [8].
2. O processamento da informação ocorre em muitos elementos simples, conhecidos como neurônios [35].
3. As forças de conexão interneurônios, denominadas sinapses ponderadas, são utilizadas para armazenar o conhecimento adquirido [8].
4. Os sinais são passados entre os neurônios através das sinapses [35].
5. Cada sinapse tem um peso associado, o qual, nas rede neurais típicas, multiplicam o sinal transmitido [35].
6. Cada neurônio aplica uma função de ativação (geralmente não-linear) a entrada da rede (soma dos sinais de entrada ponderados) para determinar seu sinal de saída [35].

As RNAs são desenvolvidas com o objetivo de criar modelos matemáticos capazes de generalizar um domínio de conhecimento específico, dando-nos a condição de gerar um processo artificial de aquisição do conhecimento ou aprendizagem. A rede passa por um processo de aprendizagem, onde ocorre a atualização das sinapses, de forma ordenada para o alcance de um objetivo de projeto desejado. Os métodos de aprendizagem têm um papel importante no desempenho das RNAs, pois por meio deles são dadas condições para realizar uma medida de representação daquilo que a rede deve aprender, e os parâmetros da rede são otimizados em relação a esta tarefa [8].

Na literatura podemos encontrar diversos modelos de RNA. Em geral, consideramos que estes modelos podem ser caracterizados e diferenciados de acordo com:

1. a topologia ou arquitetura da rede, que corresponde ao padrão de conexão entre os neurônios;
2. o método de atualização dos pesos das sinapses, denominado treinamento, aprendizagem ou algoritmo;

No estudo e desenvolvimento de RNA é comum visualizarmos os neurônios arranjados em camadas. O padrão deste arranjo de neurônios em camadas e os padrões de conexões entre as camadas é denominado arquitetura ou topologia da rede. Em geral, os neurônios que estão numa mesma camada apresentam o mesmo comportamento, determinado por suas funções de ativação por seus padrões de ponderação das

conexões, através das quais serão encaminhados e recebidos os sinais que trafegarão pela rede. Haykin [8] agrupa as arquiteturas das RNA em três grupos: redes alimentadas adiante com camada única; redes alimentadas diretamente com múltiplas camadas; e redes recorrentes. As redes alimentadas adiante com camada única tratam-se de redes neurais com uma camada de entrada, que recebe os dados do ambiente externo, a qual é projetada sobre uma camada de saída de neurônios, cuja alimentação ocorre necessariamente nesta ordem. O segundo grupo é diferenciado pela existência de uma ou mais camadas ocultas de neurônios. As camadas das redes com esta topologia estão interconectadas de tal forma que os dados oriundos da entrada sejam projetados para a camada oculta e desta para a camada de saída. Em caso de existência de mais de uma camada de oculta, os sinais da rede são projetados da entrada, passando por cada camada oculta, até chegar a saída. As redes recorrentes apresentam uma ou mais conexões conhecidas como laços de realimentação. Segundo Haykin, um sistema dinâmico possui realimentação sempre que a saída de um elemento do sistema influencia à entrada aplicada àquele elemento particular. Este tipo de rede também podem possuir uma ou mais camadas ocultas.

Além da arquitetura, o processo de atualização dos pesos das sinapses que conectam os neurônios da rede é outra característica fundamental que diferencia os modelos de redes neurais. Este processo de atualização é conhecido como treinamento ou aprendizagem da rede neural. O treinamento da uma rede neural é um processo iterativo, divido em épocas, o qual, segundo Haykin [8], possui as seguintes etapas:

1. a rede neural é estimulada por um ambiente. Em cada época do treinamento, todos os dados (estímulos) de entrada disponíveis são apresentados à rede;
2. os parâmetros da rede são adaptados, como resultado dos estímulos recebidos, efetuando ajustes nos pesos das sinapses ponderadas, de acordo com o tipo de processo de aprendizagem definido pelo modelo de rede neural que está sendo gerado;
3. a rede responde de uma nova maneira ao ambiente, em razão das modificações que foram efetuadas pelo recebimento dos estímulos.

O número de épocas executadas influenciará diretamente no resultado final que será alcançado pela rede. É durante a execução do treinamento que a rede irá construir e adaptar um modelo matemático que tem como meta generalizar um domínio de problema para o qual a rede está sendo treinada, armazenando nas estruturas topológicas o conhecimento que está sendo adquirido. Define-se que uma boa generalização da rede é alcançada quando ela é capaz de executar um mapeamento de entrada-saída corretamente (ou aproximadamente correto) para dados de teste não utilizados para a criação ou treinamento da rede.

Segundo Haykin [8], o tipo de processo de aprendizagem é definido levando em consideração a forma pela qual os parâmetros da rede são atualizados. Os processos de aprendizagem podem ser agrupados de acordo com a regra que seguem para atualizar os parâmetros livres da rede ou de acordo com o paradigma utilizado. As regras dizem respeito à maneira como ocorrerá a interpretação dos estímulos recebidos pela rede e como os ajustes dos parâmetros da rede serão executados. Os paradigmas determinam a maneira pela qual o processo de generalização será alcançado. Se com o auxílio de um professor, se com avaliações estatísticas dos estímulos recebidos, sem o auxílio externo, ou se com indicações que reforçam ou penalizam o resultado alcançado.

As regras básicas que separam os processos de aprendizagem são: aprendizagem por correção de erro, aprendizagem baseada em memória, aprendizagem hebbiana, aprendizagem competitiva e aprendizagem de Boltzmann. A diferença básica entre estas regras é maneira que aplicam o ajustes nas sinapses ponderadas da rede neural. A aprendizagem por correção de erro considera o sinal recebido, calcula um sinal de saída e a diferença ao sinal desejado, denominada erro. Uma função que minimiza este erro deve ser aplicada a fim de se aproximar do sinal desejado. Na aprendizagem basada em memória todas (ou praticamente todas) as experiências passadas pela rede são armazenadas, criando um repositório de exemplos de entrada-saída classificados corretamente. A aprendizagem hebbiana considera que se dois neurônios em ambos os lados de uma sinapse são ativados simultaneamente, então a força daquela sinapse é seletivamente aumentada. Caso os dois neurônios sejam ativados em momentos distintos, então a força da sinapse é reduzida ou eliminada. Na aprendizagem competitiva os neurônios competem entre si para se tornar ativos. Neste tipo de aprendizado somente um neurônio pode estar ativo ao mesmo tempo. O processo competitivo permite a descoberta de características estatísticas que poderão ser utilizadas para executar a tarefa de classificação dos dados apresentados à rede. O aprendizado de Boltzmann é um processo estocástico baseado na mecânica estatística. Haykin [8] apresenta formalmente estes processos de aprendizagem.

Com relação aos paradigmas, podemos separar os processo de aprendizagem em aprendizagem com um professor e aprendizagem sem um professor. A aprendizagem com um professor, também conhecida como **aprendizagem supervisionada**, é executada apresentando uma sequência de padrões de entrada (ou vetores de treinamento) associados a rótulos (*label*) indicativos das saídas desejadas para cada padrão. Sabemos que dados reais normalmente não possuem amostra para treinamento com dados pré-classificados (rótulo de classe), que são obrigatórios para execução de algoritmos de aprendizagem supervisionado. Segundo Matsubara [36], essa é uma das maiores restrições do aprendizagem supervisionado. Com o objetivo de suprir essa falta, constrói-se um modelo que agrupa os dados a partir de uma **aprendizagem não-supervisionada** ou auto-organizada, executando-se um agrupamento sem a necessidade de dados previamente classificados [37] ou sem o auxílio de um professor. Segundo Wang *et al.* [38], agrupamento é o processo de agrupar um conjunto de dados de determinada forma, na qual a similaridade entre os dados de um grupo é maximizada enquanto a similaridade entre dados de grupos diferentes é minimizada. Haykin [8] afirma que os algoritmos para aprendizagem não-supervisionada têm como objetivo descobrir padrões significativos ou características nos dados de entrada. A alteração com relação aos algoritmos supervisionados está no fato da ausência de um agente externo no papel de professor ou crítico que supervisionará o processo de aprendizagem. Por meio de um conjunto de regras, o algoritmo é capaz de aprender a calcular um mapeamento de entrada-saída com propriedades desejáveis específicas do conjunto de dados apresentado ao algoritmo. A **aprendizagem por reforço**, outro tipo de aprendizagem sem um professor, é executada por meio de recompensas que fornecem uma indicação de que o comportamento do sistema é desejável ou indesejável.

Nas Subseções 4.1.1 e 4.1.2 apresentamos os conceitos e características dos Mapas Auto-Organizáveis (SOM, do inglês *Self-Organizing Map*), uma RNA baseada em aprendizagem auto-organizável competitiva proposto por Teuvo Kohonen [39] e também da rede de Aprendizagem por Quantização Vetorial (LVQ, do inglês *Learning Vector Quantization*), outro modelo de RNA baseado em aprendizagem supervisionada competitiva também proposto por Teuvo Kohonen.

4.1.1 SOM

Segundo Han *et al.* [40], SOM é um tipo de RNA baseada em aprendizagem auto-organizável, que é similar ao aprendizagem da rede neural biológica no cérebro humano. Sua característica principal é alterar os parâmetros da rede por meio da auto-organização, que é alcançada buscando, entre seus neurônios, as melhores representações dos dados de entrada e, executando automaticamente, a atribuição intrínseca das características dos exemplos apresentados à rede, ou seja, durante o processo de aprendizagem, à medida que os dados de entrada são apresentados, suas características são utilizadas para modificar as configurações da rede (posição dos neurônios de saída no espaço vetorial e na vizinhança topológica), a qual irá adaptar-se automaticamente ao conjunto de dados.

Costa [7] relata que a visão moderna do SOM é de uma ferramenta computacional para visualização de dados. Como os dados de aplicações reais são, geralmente, multidimensionais, torna-se difícil a visualização das relações existentes entre os dados. Haykin [8] relata que o principal objetivo do SOM é mapear um padrão de entrada de dimensão arbitrária em um reticulado (*grid*) discreto uni- ou bidimensional e realizar este mapeamento adaptativamente de uma maneira topologicamente ordenada.

Um SOM consiste de reticulado de neurônios arranjados topologicamente em uma vizinhança arbitrária, denominado neste documento de camada U. As entradas da rede são vetores no espaço p-dimensional, geralmente $\Re^p$. Cada neurônio i da camada U possui um vetor associado, também no espaço $\Re^p$, $\vec{m_i}= [m_{1i}, m_{2i}, ..., m_{pi}]$.

Os neurônios da camada U são conectados aos neurônios adjacentes por uma relação de vizinhança, representada por matriz ou vetor que forma o reticulado (*grid*) dos neurônios, que descreve a estrutura do mapa ou estrutura topológica [7]. A Figura 4.1 ilustra um SOM para o caso em que $p = 3$, e a camada U possui um reticulado tamanho 7 x 10.

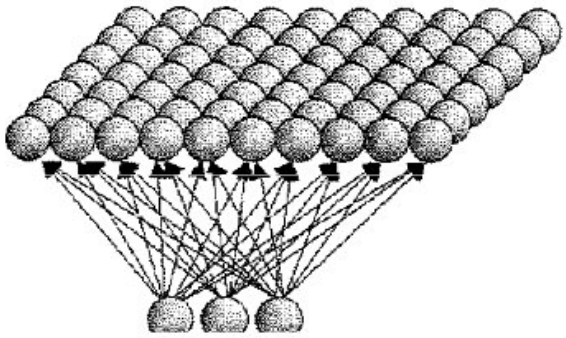
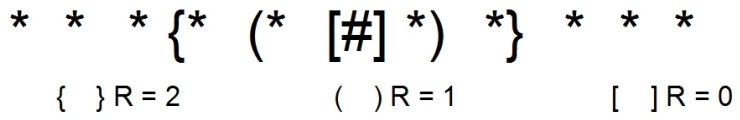


Figura 4.1: SOM com tamanho 7x10 e dimensionalidade de entrada $p = 3$ [7].

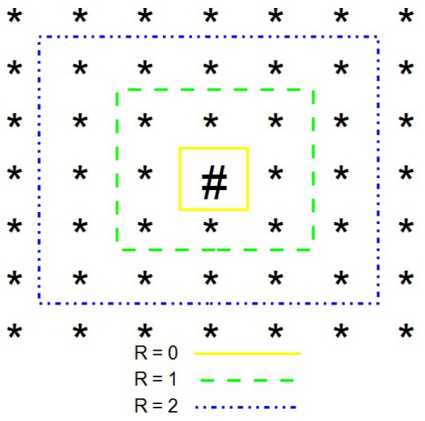
O processo de aprendizagem auto-organizável executado pelo SOM é definido como aprendizagem competitivo baseado em distância vetorial, no qual a unidade de saída que melhor representa o dado de entrada é escolhida a vencedora da competição e é denotada como *Best Matching Unit* (*BMU*). Segundo Haykin [8], na aprendizagem competitiva os neurônios de saída de uma rede neural competem entre si para tornarem-se ativos (disparar). Neste tipo de aprendizagem apenas um neurônio pode estar ativo em determinado instante. O mesmo autor acrescenta que essa característica torna a aprendizagem competitiva mais

adequada para descobrir características estatisticamente salientes que podem ser utilizadas para classificar um conjunto de padrões de entrada.

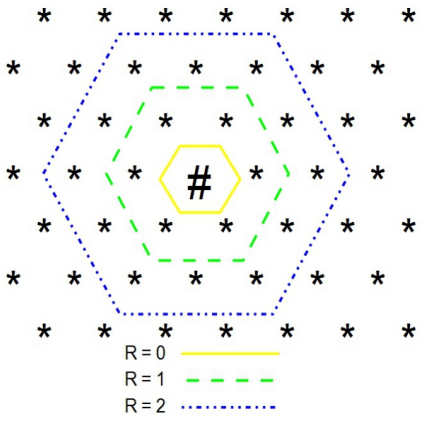
Para o alcance do objetivo proposto pelo SOM, descrito anteriormente, são consideradas dois tipos de distâncias [7]. A primeira diz respeito à distância no espaço vetorial, ou espaço de pesos, que é obtida entre um padrão de entrada $\vec{x_j}$ e um neurônio $\vec{m_i}$. O cálculo desta distância dependerá do tipo do problema, mas, em geral, usa-se a Distância Euclidiana. Durante o processo de competição define-se a BMU utilizando esta medida de distância. A outra diz respeito à distância no espaço matricial, isto é, no reticulado que descreve as relações de vizinhança dos neurônios da camada U do SOM. Ela é tomada considerando os índices dos vetores que compõe o reticulado, (k, c), onde k e c representam as coordenadas dos neurônios no *grid*. Por meio da Figura 4.2 Fausett [35] ilustra alguns exemplos de vizinhança topológica. Considere R como sendo raio de vizinhança topológica, ou seja, o quão distante os neurônios vizinhos estão, no espaço matricial, da BMU. Os símbolos $\#$ e $*$ representam, respectivamente, a BMU e seus vizinhos topológicos.



(a) Topologia unidimensional



(b) Topologia retangular



(c) Topologia hexagonal

Figura 4.2: Estruturas de vizinhanças topológicas. Adaptado de [35].

4.1.1.1 Processo de Treinamento

Define-se o treinamento do SOM, com a execução de cinco processos:

1. **Inicialização da rede**: de acordo com Haykin [8], o algoritmo responsável pela formação do mapa auto-organizável começa com a inicialização aleatória dos pesos sinápticos dos neurônios $\vec{m_i}, i = 1, 2, 3, ..., c$, onde c corresponde ao número de neurônios da camada U, que compõem o reticulado. Essa inicialização aleatória fará com que nenhuma organização prévia seja imposta ao mapa.

2. **Seleção dos padrões de entrada**: Um padrão de entrada é selecionado aleatoriamente de um conjunto X de padrões (ou de dados). Bezdek [41] denota este conjunto de dados conforme Equação

4.1.

$$X = \{x_1, x_2, x_3, ..., x_n\} \subset \Re^p \tag{4.1}$$

onde n é o número de dados do conjuntos e p é o número de características descritivas de cada dado. O j-ésimo objeto representa os atributos da imagem de um tanque de guerra ou informações médicas sobre um paciente, etc. O vetor $\vec{x_j}$, é sua representação numérica, o qual é utilizado no processo de competição, descrito em seguida. $\vec{x_{kj}}$ é a k-ésima característica associada ao objeto j.

3. **Competição entre os neurônios**: Para cada padrão de entrada $\vec{x_j}, j = 1, 2, 3, ..., n$, é calculado o valor de uma função discriminante ou função de ativação, em relação a cada neurônio $\vec{m_i}$. Essa função fornece a base para a competição entre os neurônios. O neurônio com maior valor na função discriminante é declarada vencedor da competição (BMU). Para cálculo dessa função, geralmente, utiliza-se a Distância Euclidiana d_{ij} entre o padrão de entrada $\vec{x_j}$ e todos os neurônios $\vec{m_i}$, conforme Equação 4.2.

$$d_{ij} = \sum_{l=1}^{p} (|x_{lj} - m_{li}|^2)^{\frac{1}{2}} \tag{4.2}$$

onde p é a dimensão espaço vetorial, e $i = 1, 2, 3, ..., c$. Define-se como vencedor da competição o neurônio $\vec{m_{i^*}}$, o qual possui o menor valor d_{ij}.

4. **Cooperação**: Após encontrar-se a BMU (neurônio $\vec{m_{i^*}}$, vencedor da competição), é determinada a localização espacial de uma vizinhança topológica centrada na BMU. Essa vizinhança topológica contém um conjunto de neurônios ativados k, que será representada por $h_{k,i}$. Considerando que $d_{k,i}$ representa a distância lateral entre os neurônios i e k, assume-se que a vizinhança topológica $h_{k,i}$ é uma função unimodal da distância $d_{k,i}$, desde que duas condições sejam satisfeitas: 1 - a função atinge seu valor máximo no neurônio vencedor $\vec{m_{i^*}}$, para o qual a distância $d_{k,i}$ é zero; 2 - a amplitude da vizinhança topológica $h_{k,i}$, decresce monotonicamente com o aumento da distância lateral $d_{k,i}$, tendendo a zero quando a distância aumenta. De acordo com Haykin [8], escolha típica para $h_{k,i}$, que satisfaz estas exigência, é a *função gaussiana*, expressa na Equação 4.3, ilustrada na Figura 4.3.

$$h_{k,i} = exp(-\frac{d_{k,i}^2}{2\sigma^2}) \tag{4.3}$$

onde σ é a largura efetiva, ou raio, da vizinhança topológica, o qual será diminuído com o tempo (número de iterações). Na utilização dessa função, para determinação da vizinhança topológica, à medida que o aprendizagem ocorre, o valor de σ é reduzido, para que no processo de cooperação menos neurônios sejam ativados. Dessa maneira, a cada iteração, o raio da vizinhança topológica é reduzido, fazendo com que menos neurônios sejam influenciados no processo de adaptação sináptica, descrito a seguir. Segundo Haykin [8], uma escolha comum para redução de σ em função do tempo t é o decaimento exponencial, Equação 4.4.

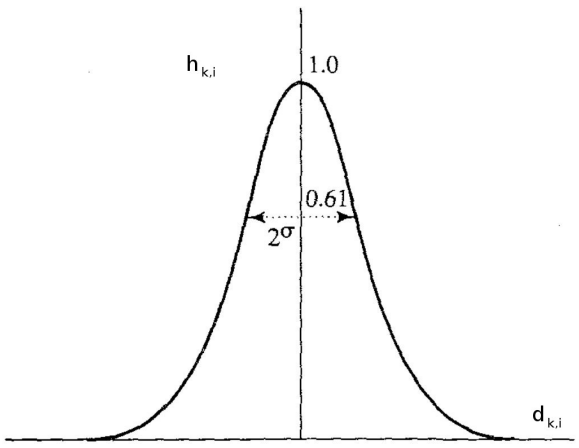


Figura 4.3: Função de vizinhança gaussiana [8].

$$\sigma(t) = \sigma_0 exp(-\frac{t}{\tau_1}), t = 1, 2, 3, ..., \tag{4.4}$$

onde σ_0 é o valor de σ na inicialização do algoritmo, τ_1 é uma constante de tempo, e t é o contador de iterações.

5. **Adaptação sináptica**: esse último mecanismo permite que os neurônios ativados (vencedor i e seus vizinhos topológicos k) aumentem seus valores individuais da função discriminante em relação ao padrão de entrada, através de ajustes adequados aplicados a seus pesos sinápticos. Esses ajustes são tais que a resposta do neurônio vencedor à aplicação subsequente de um padrão de entrada similar é melhorada. Eles são executados por meio da Equação 4.5.

$$\vec{m_i}\ (novo) = \vec{m_i}\ (antigo) + \alpha * h_{k,i} * [\vec{x_j} - \vec{m_i}\ (antigo)] \tag{4.5}$$

onde α representa a taxa de aprendizagem da rede, que decresce gradualmente a um valor pré-especificado, geralmente próximo à zero, de acordo com uma dada função de decaimento. Esse decréscimo garante o término do processo de de aprendizagem num tempo finito. Segundo Costa [7], geralmente usam-se funções monotônicas decrescentes na faixa de valores $[0, 1]$, conforme Equação 4.6.

$$\alpha(novo) = \alpha(inicial) * e^{-\frac{t}{\gamma}} \tag{4.6}$$

onde γ é o parâmetro responsável pela taxa de redução desejada e t é o número máximo de execuções que serão executadas. Por meio da Figura 4.4 Vesanto [9] ilustrou a execução do processo de adaptação sináptica na BMU e nos neurônios situados em seu raio topológico. Os círculos pretos e cinzas correspondem à posição antes e depois da atualização, respectivamente. As linhas demonstram as relações de vizinhança do mapa.

Em resumo, no processo de competição define-se a BMU. Após esta definição inicia-se o processo de cooperação, através do qual são ativados todos os neurônios dentro de um raio da BMU. Por último,

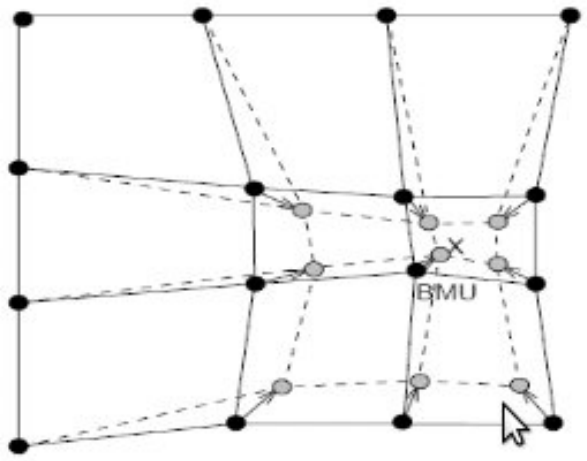


Figura 4.4: Atualização do neurônio vencedor e de seus vizinhos topológicos [9].

Algoritmo 2: Algoritmo de treinamento do SOM. Adaptado de [35].

```
// Inicialização da rede;
```
Inicialize os vetores $\vec{m_i}$ para todo i ;

Defina a estrutura topológica ;

Defina a taxa de aprendizado (α) e o raio de vizinhança (σ) inicial ;

enquanto *a condição de parada é falsa* **faça**

```
    // Seleção dos padrões de entrada;
```
    para todo *dado j (de $j = 1$ até n)* **faça**

```
        // Competição;
```
        para todo *neurônio de saída i (de $i = 1$ até c)* **faça**

            $d_{ij} = \sum_{l=1}^{p} (|x_{lj} - m_{li}|^2)^{\frac{1}{2}}$, onde p é a dimensão no espaço vetorial.

```
        // Competição;
```
        Encontre i^* tal que d_{i^*j} é mínimo ;

```
        // Cooperação e adaptação sináptica;
```
        para todo *unidade i com um raio de vizinhança específico de i^*, incluindo i^** **faça**

            $\vec{m_{i^*}}\ (novo) = \vec{m_{i^*}}\ (antigo) + \alpha * h_{k,i} * [\vec{x_j} - \vec{m_{i^*}}\ (antigo)]$

    Atualize a taxa de aprendizado ;

    Reduza o raio de vizinhança topológica ;

    Teste a condição de parada ;

executa-se a adaptação sináptica na BMU e em seus vizinhos topológicos ativados no processo anterior. Fausett [35] apresenta o Algoritmo 1 utilizado para execução do processo de treinamento do SOM.

4.1.2 LVQ

O método *Learning Vector Quantization* (LVQ), também proposto por Teuvo Kohonen, utiliza arquitetura semelhante a do SOM. A LVQ é capaz de classificar os dados por meio de aprendizagem supervisionado competitivo, no qual o neurônio de saída mais próximo do dado de treinamento é conhecido como vencedor da competição. Os demais neurônios são conhecidos como perdedores. Durante o aprendizagem, a LVQ atualiza os pesos das sinapses do neurônio vencedor de acordo com a sua classificação. Caso o dado de entrada e o neurônio de saída sejam da mesma classe (classificação correta), então os pesos das sinapses conectadas ao neurônio vencedor são atualizados de tal forma que esse se aproxima do dado apresentado. Caso contrário, o neurônio é afastado do dado. A Figura 4.5 ilustra esse processo de aprendizagem.

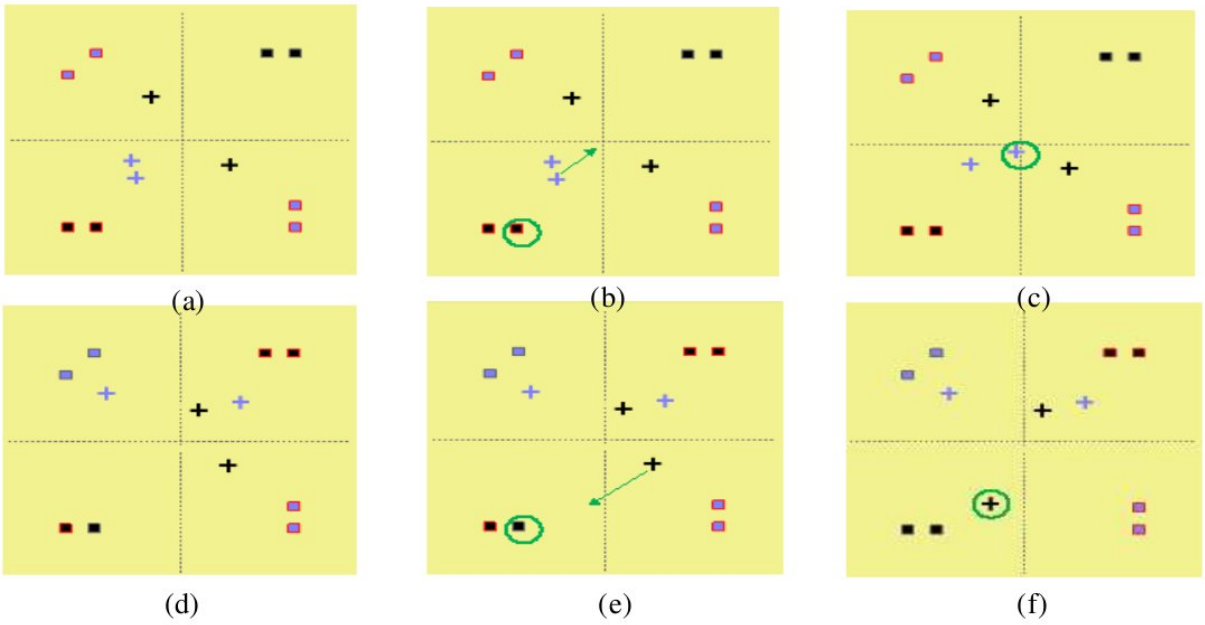


Figura 4.5: Processo de aprendizagem da rede LVQ. (a) Configuração atual da rede, onde: Quadrados: vetores de entrada (dados). Cruzes: vetores de unidades de saída (cor preta: classe 1, cor azul: classe 2). Contorno preto: classificação correta. Contorno vermelho: classificação errada. (b) Quando o dado circulado é apresentado à rede, sua classificação é feita incorretamente, o que fará com que a unidade de saída que o classificou seja afastada do dado. (c) Nova posição do vetor da unidade de saída, após o aprendizagem. (d) Configuração atual da rede. (e) Quando o dado circulado é apresentado à rede, sua classificação é feita corretamente, o que fará com que a unidade de saída que o classificou seja atraída na direção e sentido do dado. (f) Nova posição do vetor da unidade de saída, após o aprendizagem. (Toolbox NND14LV1 - MatLab ®).

Essas atualizações acontecem em função da taxa de aprendizagem atual da rede e da distância, em geral Euclidiana, entre o dado apresentado e o neurônio que venceu a competição. No decorrer das iterações t, a taxa de aprendizagem é alterada, conforme estabelecido pelo projetista da rede. Fausett [35] apresenta o

Algoritmo 2 utilizado para execução do processo de treinamento da LVQ. Considere T como a categoria ou classe correta do vetor de treinamento e C_i categoria ou classe representada pela i-ésima unidade da camada de saída.

Algoritmo 3: Algoritmo de treinamento da LVQ. Adaptado de [35].

Inicialize os vetores $\vec{m_i}$ para todo i ;
Defina a taxa de aprendizagem (α) inicial ;
enquanto *a condição de parada é falsa* **faça**
  para todo *dado j (de $j = 1$ até n)* **faça**
    para todo *neurônio de saída i (de $i = 1$ até c)* **faça**
      $d_{ij} = \sum_{l=1}^{p} \left(|x_{lj} - m_{li}|^2\right)^{\frac{1}{2}}$, onde p é a dimensão no espaço vetorial.
    Encontre i^* tal que d_{i^*j} é mínimo ;
    se $T = C_{i^*}$ **então**
      $\vec{m_{i^*}}\ (novo) = \vec{m_{i^*}}\ (antigo) + \alpha * [\vec{x_j} - \vec{m_{i^*}}\ (antigo)]$
    senão
      $\vec{m_{i^*}}\ (novo) = \vec{m_{i^*}}\ (antigo) - \alpha * [\vec{x_j} - \vec{m_{i^*}}\ (antigo)]$
  Reduza a taxa de aprendizagem ;
  Teste a condição de parada ;

4.1.3 Classificação Adaptativa de Padrões

Por meio do algoritmo de treinamento do SOM, podemos encontrar uma aproximação da divisão do espaço de entrada dos dados em diversas regiões distintas, que são representadas pelos vetores de pesos sinápticos dos neurônios do mapa de características gerado pelo SOM. Utilizando o mapa gerado e estas regiões no espaço de entrada, ao apresentarmos novos padrões de entrada (novos dados de entrada) ao mapa, somos capazes de identificar em qual região do espaço este novo dado está situado, comparando o vetor representante do dado com os vetores de pesos sinápticos dos neurônios de saída. De acordo com Haykin [8], o cálculo do mapa de características pode ser visto como o primeiro de dois estágios para resolver de forma adaptativa um problema de classificação de padrões, como ilustrado na Figura 4.6. O segundo estágio é realizado pela LVQ, que fornece um mecanismo para o ajuste fino do mapa gerado pelo SOM.

4.2 DESENVOLVIMENTO DE APLICAÇÕES UTILIZANDO REDES NEURAIS

O processo de desenvolvimento de aplicações utilizando redes neurais, ilustrado na Figura 4.7, pode ser divido nas etapas de: coleta dos dados relativos ao problema; separação do conjunto de dados; configuração da rede; treinamento; teste e integração.

Na etapa de coleta dos dados relativos ao domínio do problema devemos cobrir amplamente o domínio do problema com dados significativos em todas as possibilidade, incluindo exceções e condições nos

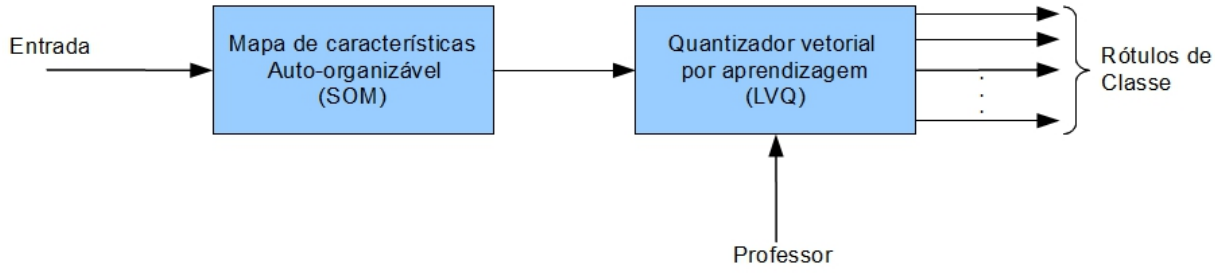


Figura 4.6: Diagrama em blocos da classificação adaptativa de padrões, usando um mapa de características auto-organizável e quantizador vetorial por aprendizagem. Adaptado de [8].

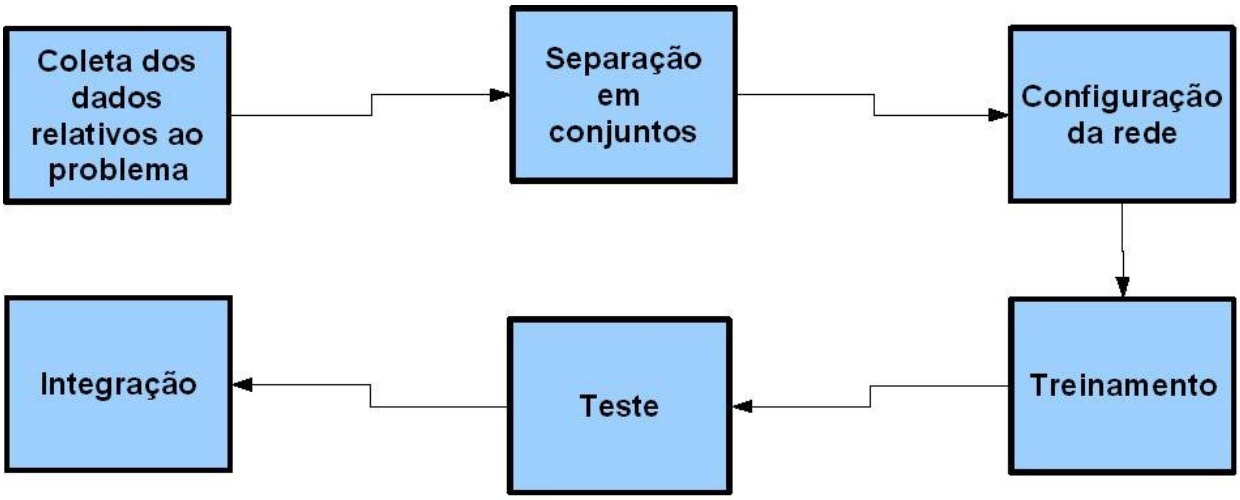


Figura 4.7: Processo de desenvolvimento de aplicações que utilizam RNA.

limites do domínio. Através destes dados a rede será treinada e ocorrerá o aprendizado.

A separação do conjunto de dados é a tarefa na qual se dividem os dados num conjunto para treinamento da rede e outro para o teste de seu desempenho. Os dados de treinamento são os dados que fornecem exemplos para o aprendizagem da rede (experiência fornecida) e os dados de teste são utilizados para avaliar o desempenho da rede sob condições diferentes das apresentadas durante o treinamento. Também é muito comum utilizar o conjunto de dados para gerar um ou mais subconjuntos de treino e de teste, para executar a validação cruzada do modelo gerado. Na Seção 6.3 tratamos deste assunto com mais detalhes.

O processo de configuração da rede envolve a seleção da arquitetura de rede apropriada à aplicação, a determinação da topologia da rede (número de camadas, número de neurônios em cada camada, conexão entre os neurônios), a determinação dos parâmetros do algoritmo de treinamento e das funções de ativação. Esta é uma etapa onde as escolhas são empíricas, requerendo experiência dos projetistas da rede.

No treinamento são definidos os critérios de inicialização da rede. Uma boa escolha para estes critérios pode diminuir o tempo necessário para o processamento. Também define-se o modo como o treinamento será executado, envolvendo a ordem de apresentação dos dados e a avaliação de necessidade de normalização dos dados. O tempo de treinamento é outro que influencia no resultado do treinamento. Em geral definem-se critérios de parada para o algoritmo, como por exemplo, quantidade de iterações (ou épocas) ou taxa de erro suficientemente pequena.

Na etapa de teste verifica-se o desempenho da rede, servindo como uma boa indicação para o desempenho da rede quando ela estiver em operação com dados reais. Após o treino e teste da rede neural efetua-se sua integração à aplicação onde será utilizada. O ideal é que o sistema monitorasse periodicamente o desempenho durante a operação, utilizando critérios de análise semelhantes aos utilizados na etapa de teste, apontando a necessidade de execução de um novo treinamento da rede.

4.3 CONSIDERAÇÕES FINAIS

O desenvolvimento de aplicações que emulam um comportamento inteligente permite que tarefas cotidianas dos homens possam ser executadas por máquinas, mesmo quando exigem o uso de raciocínio, planejamento e aprendizado. As Redes Neurais Artificiais são ferramentas robustas comumente utilizadas para resolver problemas que exigem aprendizado, como a classificação e o reconhecimento de padrão.

Neste Capítulo buscamos descrever os principais conceitos envolvidos na teoria das Redes Neurais artificiais, apresentando suas principais arquiteturas e seus principais algoritmos de aprendizado, dando ênfase nos Mapas Auto-Organizáveis (SOM, do inglês *Self-Organizing Map*) e na rede de Aprendizagem por Quantização Vetorial (LVQ, do inglês *Learning Vector Quantization*). Encerramos o Capítulo descrevendo um processo de desenvolvimento de aplicações que utilizam as RNA, apresentando os seis passos essenciais deste processo. Na coleta dos dados buscamos encontrar exemplos do domínio do problema para o qual estamos desenvolvendo a aplicação, que serão apresentados à rede para seu treino e teste. Em seguida efetuamos a separação do conjunto de dados coletado, para definirmos os dados que servirão de teste e aqueles que servirão para o treino. Os passos seguintes são a configuração da rede, o treinamento e o teste, que dependerão da arquitetura da rede e do algoritmo de aprendizado escolhidos pelo projetista da

aplicação. Por fim, executamos a integração do modelo gerado pela rede à aplicação onde ela será utilizada.

No próximo Capítulo apresentaremos a metodologia proposta para o reconhecimento de movimentos em sequências de imagens, que utiliza os Histogramas de Fluxo Óptico Orientado e a Análise de Contorno como representações do padrão de movimento, as quais serão apresentadas aos algoritmos de RNA, que irão reconhecer e classificar os padrões dos movimentos.

5 METODOLOGIA

Com o objetivo de executar o reconhecimento do tipo de ação humana realizada na sequência de imagens propomos uma metodologia, ilustrada na Figura 5.1, que utiliza uma combinação das técnicas de Visão e Inteligência Computacional, apresentadas nos Capítulos anteriores. Na etapa de Visão Computacional (VC) calculamos o Fluxo Óptico (FO) do movimento, o Histograma do Fluxo Óptico Orientado (HOOF, do inglês *Histrogram of Oriented Optical Flow*), bem como analisamos o contorno do objeto durante o movimento, para gerar representações numéricas da ações executadas nas sequências de imagens. Em seguida, os dados gerados através destas representação são utilizados como entrada para a fase de Inteligência Computacional (IC), que irá modelar os dados de entrada representados, armazenando o conhecimento obtido durante a etapa de treinamento, para torná-lo disponível para uso posterior, na execução da tarefa de classificação. As técnicas das Redes Neurais Artificiais SOM e LVQ são utilizadas na etapa de IC.

Nas Seções seguintes descrevemos as entradas e as saídas de cada uma das três etapas da metodologia, bem como os parâmetros utilizados para a configuração dos algoritmos de Visão e de Inteligência Computacional, buscando ilustrar e exemplificar as entradas, os resultados intermediários e as saídas de cada etapa.

5.1 ETAPA DE VISÃO COMPUTACIONAL

Na etapa de Visão Computacional analisamos cada sequência de imagens, buscando obter representações numéricas do movimento relacionado a ação que ocorreu nas imagens. Estas representações podem ser obtidas através dos Histogramas de Fluxo Óptico Orientados (HOOF), apresentados na Seção 2.3, e dos descritores de contorno dos objetos durante o movimento, apresentados no Capítulo 3. Considerando que ações de movimentos diferentes podem gerar o mesmo perfil de Fluxo Óptico, foi necessário utilizar os descritores de contorno para diferenciar estas ações com o mesmo Fluxo Óptico. Desta forma, as representações por HOOF e por descritores de contorno são utilizadas de maneira complementar. A entrada desta etapa é uma sequência de imagens e sua saída são representações numéricas da ação humana em cada sequência de imagens. Nas próximas Seções descrevemos como cada técnica de Visão Computacional foi aplicada na metodologia.

5.1.1 Cálculo do Fluxo Óptico e dos Histogramas Orientados de Fluxo Óptico

Conforme discussões acerca dos algoritmos de Cálculo do Fluxo Óptico contidas na Seção 2.2, notamos que o algoritmo KLT é o que apresenta maior acurácia e robustez. Os algoritmos de HS e LK consideram apenas movimentos pequenos. Caso as sequências de imagens utilizadas possuam pixels se movimentando muito rapidamente, então o cálculo das derivadas parciais irá falhar. Assim, optamos pela utilização do algoritmo KLT para o cálculo do Fluxo Óptico e escolhemos heuristicamente os seguintes parâmetros para

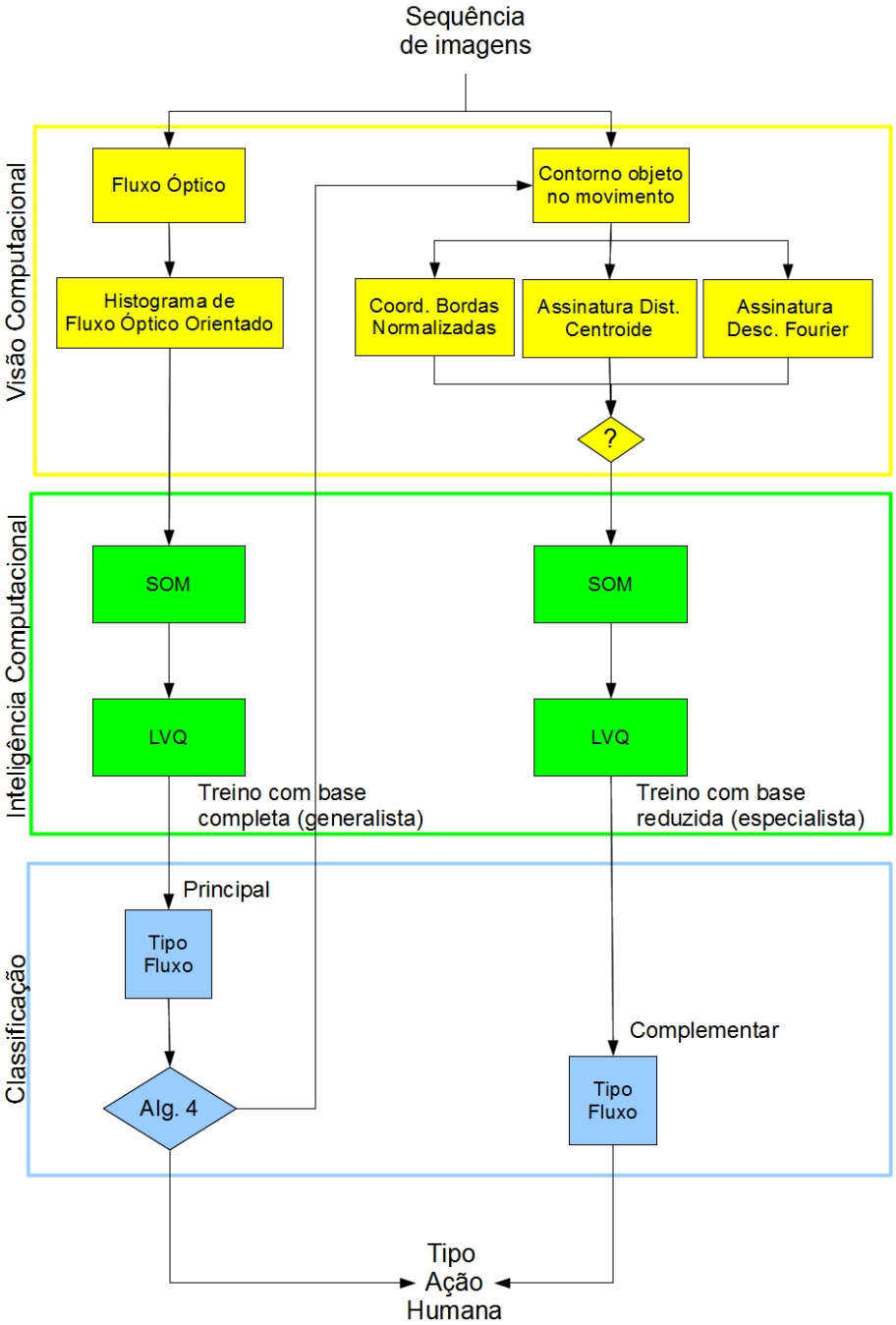
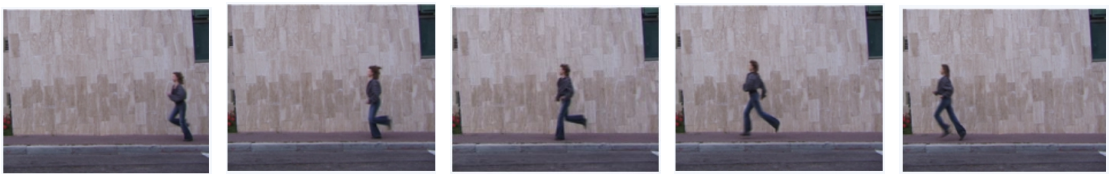


Figura 5.1: Metodologia Proposta.

a execução do algoritmo:

- Até 40.000 boas características para rastrear (*good features to track*, método proposto em [11], apresentado na Seção 2.2.4.2), com limiar de 0,01 para os autovalores dos pontos incluídos e distância mínima entre os pontos de 0,01;
- Pirâmides de imagens com cinco níveis;
- Janela de vizinhança retangular, com 5×5 pixels;
- Máximo de 20 iterações para o cálculo do Fluxo Óptico de LK.

Com o objetivo de exemplificar a entrada e a saída deste passo da metodologia, utilizamos uma sequência de imagens com 42 quadros onde está sendo executada a ação correr para frente, para calcular o Fluxo Óptico entre os quadros e seus respectivos HOOF. Apresentamos na Figura 5.2 os quadros 7, 14, 21, 28 e 35 desta sequência de imagens.



(a) 7 (b) 14 (c) 21 (d) 28 (e) 35

Figura 5.2: Quadros da sequência de imagens utilizados para exemplificar o cálculo do Fluxo Óptico e do HOOF.

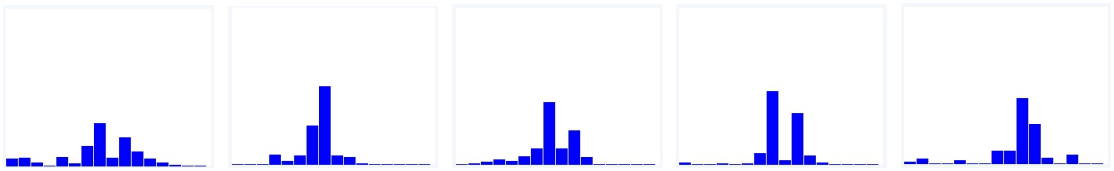
Os resultados para a execução do algoritmo proposto por Kanade, Lucas e Tomasi [10, 11] estão apresentados na Figura 5.3. Omitimos heuristicamente da Figura e do cálculo do HOOF os vetores de Fluxo Óptico que possuem módulo inferior $1,42$ e superior 15, para desprezar os vetores de módulos pequenos ou grandes demais, que podem representar fluxos não significativos ou ruídos na estimativa.

(a) 7 (b) 14 (c) 21 (d) 28 (e) 35

Figura 5.3: Cálculo do Fluxo Óptico utilizando o algoritmo de Kanade, Lucas e Tomasi [10, 11].

Na Figura 5.4 apresentamos o cálculo do HOOF com 16 *bins*, para os fluxos ópticos calculados anteriormente e na Figura 5.5 o histograma médio dos HOOF para a sequência de imagens completa.

Por meio do HOOF médio obtemos um primeiro tipo de representação numérica da ação humana ocorrida na sequência de imagens, a qual será utilizada como entrada da etapa de Inteligência Computacional. Como os HOOF são representações da distribuição de probabilidade dos ângulos primários dos vetores de Fluxo Óptico estimados, os dados de saída serão matrizes linha (ou vetores) contendo números reais



(a) 7 (b) 14 (c) 21 (d) 28 (e) 35

Figura 5.4: Cálculo do HOOF com 16 *bins*, utilizando o algoritmo de Kanade, Lucas e Tomasi [10, 11].

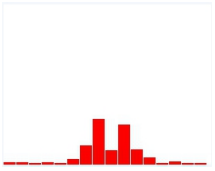


Figura 5.5: Cálculo do HOOF médio com 16 *bins*.

com valores no intervalo $[0, 1]$, cuja soma é igual a um. A dimensão do vetor ou quantidade de colunas da matriz linha dependerá da quantidade de *bins* atribuídas na geração dos HOOF.

5.1.2 Análise do contorno do objeto durante o movimento

Assim como no caso do cálculo do Fluxo Óptico, o passo de análise do contorno do objeto durante o movimento recebe como entrada uma sequência de imagens e apresenta como saída uma representação numérica da ação ocorrida. A análise do contorno requer inicialmente a extração do movimento do objeto na sequência de imagens, conforme método apresentado na Seção 3.1, a qual irá gerar uma imagem binária contendo um objeto representante do movimento. Em seguida, calculamos a posição das coordenadas de borda deste objeto utilizando o algoritmo *Moore-Neighbor Tracing*, considerando a conectividade-8 e o critério de parada de *Jacob* na sua execução, conforme discutido na Seção 3.4. Para permitir a comparação das coordenadas de bordas de diferentes sequências de imagens, bem como de suas respectivas assinaturas, executamos a normalização das bordas, de acordo com o processo apresentado na Seção 3.6. Desta forma, para executar este passo utilizamos os seguintes parâmetros:

- limiar $= 0.1$ no passo 3 do método apresentado na Seção 3.1;
- bordas normalizadas com 100 pontos;
- para calcular as assinaturas, o ponto em comum escolhido foi aquele que apresentou a menor distância ao canto superior esquerdo da imagem.
- os Descritores de Fourier foram calculados utilizando a assinatura de centroide, e foram transformados de forma a serem invariantes à escala, translação e rotação.

Com o objetivo de exemplificar os resultados intermediários obtidos nesta etapa, com a execução dos algoritmos de Rastreamento de Contorno, Normalização de Borda, Assinatura de Contorno e Descritores de Fourier, utilizamos a Figura 5.6 a qual representa a extração do movimento do objeto durante a execução da ação correr para frente, obtida com a aplicação do método apresentado na Seção 3.1.

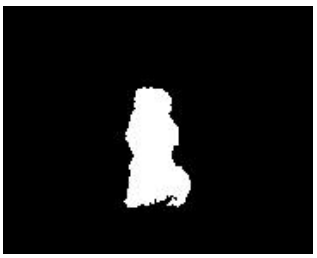


Figura 5.6: Extração do movimento do objeto durante a ação correr.

Na Figura 5.7 apresentamos o resultado do rastreamento de contorno, que teve como entrada a imagem ilustrada na Figura 5.6, utilizando o algoritmo *Moore-Neighbor Tracing*, considerando a conectividade-8 e o critério de parada de *Jacob* na sua execução. A quantidade de pontos resultante do rastreamento do contorno é dependente do tamanho e da forma do objeto ilustrado na imagem. No caso ilustrado a borda possui 197 pontos.

Figura 5.7: Contorno da imagem da Figura 5.6, utilizando o algoritmo *Moore-Neighbor Tracing*, com conectividade-8 e critério de parada de *Jacob*, com 197 pontos.

Como a quantidade de pontos de borda pode variar a depender do objeto utilizado, para permitir a comparação entre as representações dos contornos, executamos a normalização da borda, utilizando o processo de normalização por igualdade de comprimento de arco, pois ele apresenta o melhor efeito de distribuição igualitária dos pontos pela borda do contorno. Na Figura 5.8 apresentamos o resultado da normalização, considerando 100 pontos de coordenadas de borda.

Figura 5.8: Normalização da borda ilustradas na Figura 5.7, utilizando 100 pontos.

Para calcular a assinatura de contorno, baseada na distância do centroide, devemos localizar o centroide e um ponto de partida em comum em todas as bordas normalizadas, bem como definir o sentido em que a borda será percorrida para produzir a assinatura. Na Figura 5.9 o ponto vermelho representa a posição do centroide da borda normalizada com 100 pontos. O ponto em comum escolhido foi aquele que apresentou a menor distância ao canto superior esquerdo da imagem. Os pontos verde e preto indicam, respectivamente,

o primeiro e segundo ponto do sentido que a borda será percorrida para gerar a assinatura.

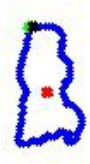


Figura 5.9: Localização do centroide (ponto vermelho) da borda normalizada com 100 pontos e do ponto de partida (ponto verde) para cálculo da assinatura baseada na distância do centroide, que será obtida no sentido do primeiro (verde) para o segundo ponto (preto).

Na Figura 5.10 apresentamos a Assinatura da Distância do Centroide para a borda normalizada com 100 pontos, cujos centroide, ponto de partida e sentido estão ilustrados na Figura 5.9. As representações de contorno serão diferentes para cada tipo de ação, razão pela qual estas representações podem ser utilizadas por algoritmos de reconhecimento de padrões, como as redes neurais artificiais.

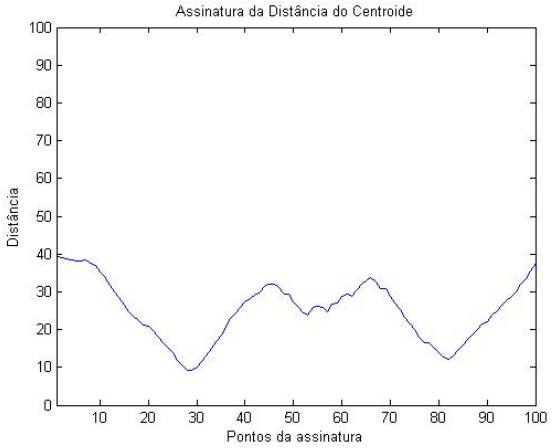


Figura 5.10: Assinatura da Distância do Centroide, com a borda normalizada em 100 pontos.

Na Figura 5.11 apresentamos os padrões gerados pelos módulos dos Descritores de Fourier invariantes à Translação, Rotação e Escala da assinatura da Distância do Centroide.

Considerando os três tipos de representações de contorno ilustrados nas Figuras anteriores (coordenadas de bordas normalizadas, assinatura da distância do centroide e módulo dos descritores de Fourier invariantes a translação, rotação e escala) e observando os padrões gerados para outras ações, não ilustrados neste documento, percebemos que estas representações são capazes de gerar um padrão para cada tipo de ação. Estes padrões, que possuem natureza numérica, podem ser utilizados como entrada para algoritmos de reconhecimento de padrões, como as Redes Neurais Artificiais apresentadas no Capítulo 4. Para a representação que utiliza as coordenadas dos pontos de bordas teremos uma matriz 100×2 para cada sequência de imagem. Os valores desta matriz são naturais e se referem às posições dos pixels de borda na imagem. No caso da representação que considera os descritores baseados na assinatura da distância do centroide a saída será do tipo vetorial com 100 dimensões. Os valores dos vetores são reais positivos que

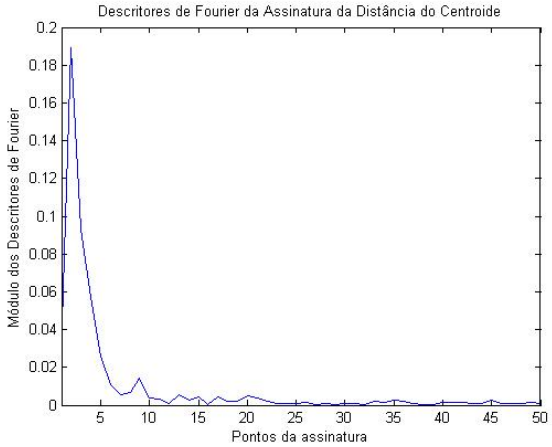


Figura 5.11: Módulo dos Descritores de Fourier invariantes a Translação, Rotação e Escala da Assinatura da Distância do Centroide, com a borda normalizada em 100 pontos.

indicam a distância euclidiana do ponto de borda ao ponto centroide. Por fim, os módulos dos descritores Fourier serão vetores reais com 50 dimensões, já que a transformação para torná-lo invariante à translação, rotação e escala reduz a quantidade de descritores pela metade.

Os fatores que distinguem os tipos de representações, que influenciarão no desempenho dos algoritmos de reconhecimento de padrões, são: o nível de detalhes que a representação é capaz de mapear, a capacidade de minimizar a semelhança entre padrões que representem objetos diferentes e a dimensionalidade do padrão. Os dois primeiros fatores fornecerão ao algoritmo de reconhecimento de padrões informações para o aprendizado. O último fator irá aumentar ou reduzir o esforço computacional necessário para que o aprendizado ocorra. O ideal é buscar encontrar um ponto de equilíbrio na escolha dos valores destes fatores.

Especificamente para o caso da representação do contorno pelas coordenadas da borda normalizada, ilustrada na Figura 5.8, o nível de detalhes mapeado, a dimensionalidade do padrão e a minimização da semelhança entre padrões que representam objetos diferentes dependerão da quantidade de pontos definidos na normalização. Considerando os demais tipos de representação, entendemos que como eles são obtidos com base na representação por bordas normalizadas, eles apresentarão menos informações ao algoritmo de reconhecimento de padrões, o que poderá prejudicar o desempenho do algoritmo.

5.2 ETAPA DE INTELIGÊNCIA COMPUTACIONAL

A etapa de Inteligência Computacional recebe como entrada as representações numéricas dos movimentos ocorridos na execução das ações humanas. Estas representações foram obtidas quando da execução da etapa de Visão Computacional. Assim, suas possíveis entradas são:

1. representação do Fluxo Óptico por meio do HOOF médio da sequência de imagens;
2. representação do contorno do movimento do objeto na sequência de imagens obtida:

- por meio das coordenadas de bordas normalizadas ou;
- por meio da assinatura de contorno baseada na distância do centroide ou;
- por meio da assinatura do contorno baseado nos Descritores de Fourier.

Cada uma delas possui suas respectivas vantagens e desvantagens e sua robustez foi avaliada na execução dos experimentos apresentados no Capítulo 6. Todas as representações buscam gerar padrões de entrada que poderão ser classificados pelos algoritmos das RNAs.

A saída desta etapa são classificadores neurais capazes de generalizar o domínio do problema (tipo de ação humana). Esta modelagem é obtida após a execução do processo de treinamento da rede neural, que busca armazenar nas estruturas internas da rede as características dos exemplos de entrada apresentados. Assim, os classificadores gerados poderão classificar ações humanas executadas em sequências de imagens não utilizadas durante o treinamento.

5.2.1 Parâmetros para as Redes Neurais Artificiais

Nestes experimentos utilizamos a classificação adaptativa de padrões (SOM+LVQ). Nela a posição dos neurônios calculada pelo SOM é utilizada para inicializar a LVQ.

Boscarioli [42] elencou as principais estratégias para a parametrização do SOM. Cabe destacar que não existem critérios bem definidos para a escolha dos valores de tais parâmetros e sim heurísticas que podem contribuir para a geração de mapas que produzam a melhor solução. Essas estratégias são:

- **Encontrar a dimensão ideal para o mapa.** Para análise dos dados, em geral utilizam-se mapas bidimensionais. Peres [43] apresentou estudo acerca da escolha de outras dimensões para o mapa.
- **Encontrar o número ideal de neurônios.** A quantidade de neurônios determinará a granularidade do resultado do mapeamento, afetando o custo computacional, a qualidade e a capacidade de generalização do SOM. Boscarioli cita duas heurísticas para esta escolha: 1 - sendo n o tamanho do conjunto de dados, então, a quantidade de neurônios do mapa (q) será $q = 5\sqrt{n}$; 2 - caso o tamanho do conjunto de dados seja inferior a 1.000, então $q = n$.
- **Definir o formato do mapa.** Os mapas podem estar no formato hexagonal, retangular, cilíndrico ou toroidal.
- **Definir a função de vizinhança.** Além da função de gaussiana (Equação 4.3), outras funções podem ser escolhidas, entretanto, a função gaussiana é a mais utilizada pois, normalmente, tem melhor tendência de agrupamento.
- **Inicialização dos vetores de pesos (neurônios de saída ou protótipos.** A inicialização pode ser feita de três formas: 1 - aleatoriamente, onde os vetores protótipos são iniciados em posições aleatórias do espaço vetorial; 2 - com exemplos, onde exemplos são retirados aleatoriamente do conjunto de dados de entrada para definição da posição dos vetores protótipos na espaço vetorial; 3 - com base em distribuições estatísticas dos dados de entrada, utilizando conceitos de componentes principais, autovalores, autovetores e autocorrelação, que irá impor alguma ordem já na inicialização do mapa.

Seguindo estas estratégias, utilizamos os seguintes parâmetros para execução do SOM:

- **Convergência da rede**: foram utilizados dois critérios para definir que ocorreu a convergência da rede: a soma das movimentações de todos os neurônios de saída (ϵ) e número de iterações máximo (t_{max}). Caso ϵ fosse inferior ou igual ao limite estabelecido para cada experimento, ou a quantidade de iterações superior ou igual a t_{max} a execução do treinamento era interrompida. Foram considerados $t_{max} = 100$ e $\epsilon = 0.08$.
- **Topologia do Mapa e Quantidade de Neurônios**: Foi utilizada a topologia retangular. Para definir a quantidade de neurônios seguimos um valor próximo ao sugerido por Boscarioli [42]. Utilizamos 81 neurônios, com topologia 9×9 para as redes dos experimentos do Tipo 1 e também para as redes generalistas dos experimentos do Tipo 2 e 3. Para as redes especialistas dos experimentos do Tipo 2 utilizamos 16 neurônios, com topologia 4×4, e para as do Tipo 3 utilizamos 30 neurônios, com topologia 6×5.
- **Vizinhança topológica**: foi utilizada a função gaussiana (Equação 4.3) para determinação da vizinhança topológica e o decaimento do raio foi exponencial, conforme Equação 4.4. O valor inicial do raio foi configurado em 2.
- **Ordem dos dados apresentados**: todos os dados foram apresentados com ordem aleatória.
- **Pesos das sinapses**: os pesos das redes foram iniciados com valores aleatórios entre 0 e 1.
- **Taxa de aprendizagem**: Foram utilizados valores entre 0 e 1 e o decaimento de α durante o treinamento foi monotônico, seguindo o estabelecido na Equação 4.6. No caso do SOM foi considerado $\alpha_{inicial} = 0.9$.
- **Medidas de Distância**: todas as medidas de distância foram tomadas considerando a Distância Euclidiana, definida na Equação 4.2.

Para a etapa da LVQ utilizamos:

- **Convergência da rede**: também foram utilizados dois critérios de parada da para LVQ: o número de iterações máximo (t_{max}) e a taxa de acerto na classificação alcançada. Foram considerados $t_{max} = 30$ e a taxa variou em cada experimento.
- **Pesos das sinapses**: Os pesos da LVQ foram inicializados com os pesos dos vetores gerados pelo mapa de características do SOM.
- **Taxa de aprendizagem**: Foram utilizados valores entre 0 e 1 e o decaimento de α durante o treinamento foi monotônico, seguindo o estabelecido na Equação 4.6. No caso da LVQ $\alpha_{inicial} = 0.3$
- **Ordem dos dados apresentados**: todos os dados foram apresentados com ordem aleatória.
- **Medidas de Distância**: todas as medidas de distância foram tomadas considerando a Distância Euclidiana, definida na Equação 4.2.

A normalização dos dados apresentados às redes também é um parâmetro que influencia diretamente nos resultados obtidos. Quando utilizamos o HOOF como dados de entrada das redes, não foi necessário executar a normalização dos dados. Nos demais casos, a normalização foi executada.

5.3 ETAPA DE CLASSIFICAÇÃO

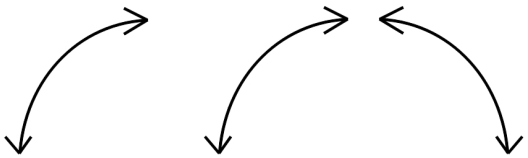
A etapa de classificação recebe como entrada matrizes que representam as estruturas internas das RNA, as quais foram geradas durante os treinamentos executados na fase de IC e apresenta como saída a definição do tipo de ação humana efetuada na sequência de imagens. Como pode ser observado na Figura 5.1, a classificação pode ser executada em até duas etapas. A primeira classificação efetuada define o tipo de Fluxo Óptico que está ocorrendo na sequência de imagens. Considerando que a estimativa de Fluxo Óptico e os Histogramas de Fluxo Óptico Orientado (HOOF) implementam representações compactas das características do movimento, as representações geradas podem ser semelhantes para movimentos distintos, como por exemplo as ações acenar com um braço e acenar com dois braços ilustradas na Figura 5.12.

(a) (b)

Figura 5.12: Exemplo de movimentos distintos que possuem a mesma representação de Fluxo Óptico. (a) acenar com um braço. (b) acenar com dois braços.

Estas ações possuem fluxos ópticos distintos. O fluxo da primeira ação se assemelha a um semi-arco bidirecional e a segunda a dois semi-arcos bidirecionais, conforme ilustrados na Figura 5.13. Apesar dos fluxos serem diferentes, ambos possuem o mesmo tipo de representação por HOOF, já que a distribuição de probabilidade dos vetores por histogramas desconsidera o sentido e a escala do fluxo, que é efetuada com base nos valores dos ângulos primários dos vetores. Considerando esta limitação da representação utilizada e considerando também que o conjunto de treino das RNA é composto por elementos conhecidos, na etapa de Inteligência Computacional deveremos agrupar no conjunto de treinamento as ações diferentes que possuem a mesma representação de HOOF. Desta maneira, elas serão classificadas com o mesmo tipo de Fluxo Óptico. A necessidade deste agrupamento pode ser avaliada conforme procedimento descrito no Algoritmo 4.

Neste trabalho consideramos $l = 0.07$ e obtemos indicativo de agrupamento das classes acenar com dois braços e acenar com um braço e também das classes correr e pular para frente, indicando que estas



(a) (b)

Figura 5.13: Ilustração dos fluxos ópticos das ações humanas apresentadas na Figura 5.12. (a) Fluxo Óptico para o movimento acenar com um braço. (b) Fluxo Óptico para o movimento acenar com dois braços

Algoritmo 4: Algoritmo para avaliação da necessidade de agrupamento das classes de movimento presentes no conjunto de treinamento.

Entrada: Conjunto de treinamento contendo n HOOF referentes à c classes de movimentos. Conjunto deverá possuir *label* cl indicativo da classe do movimento executado na sequência de imagens.

Saída: Classes similares, que podem ser agrupadas.

Defina um limiar l que caracterizará as classes como semelhantes, atribuindo a ele um valor real no intervalo $[0, 1]$. Quanto mais próximo de zero, maior será a similaridade ;

para todo *classe* i *(de* $i = 1$ *até* c*)* **faça**

  Calcule o histograma médio $\bar{h_i}$ da classe i, conforme Equação 2.45, considerando todo HOOF com $cl = i$;

para todo *classe* i *(de* $i = 1$ *até* c*)* **faça**

  para todo *classe* j *(de* $j = 1$ *até* c*)* **faça**

    Calcule $d_{\chi^2}(\bar{h_i}, \bar{h_j})$, a distância χ^2 entre os histogramas médios, conforme:

    $d_{\chi^2}(\bar{h_i}, \bar{h_j}) = \frac{1}{2} \sum_{b=1}^{B} \frac{\left|\bar{h_{1;b}} - \bar{h_{2;b}}\right|^2}{\bar{h_{1;b}} + \bar{h_{2;b}}}$;

    se $d_{\chi^2}(\bar{h_i}, \bar{h_j}) < l$ **então**

      Inclua as classes i e j como candidatas ao agrupamento ;

classes possuem perfil de Fluxo Óptico semelhante e o processo de classificação utilizando apenas a técnica de Fluxo Óptico não seria capaz de separa estas classes apresentando bons resultados.

Por este motivo, após esta fase, caso a ação seja classificada como pertencente ao tipo de Fluxo Óptico que possui uma ou mais ações distintas, então os elementos que foram inicialmente agrupados são desagrupados de acordo com a classe de ação humana original, para gerar um conjunto de treinamento menor que irá produzir uma rede neural especializada nas ações deste tipo de Fluxo Óptico. Esta rede especialista utiliza apenas a representação obtida na análise de contorno do objeto durante o movimento.

Com este processo de classificação executado em até duas etapas garantimos a classificação das ações humanas. Caso dois ou mais movimentos distintos possuam a mesma representação de Fluxo Óptico, utilizamos, de maneira complementar, a etapa de classificação do movimento por tipo de contorno. Caso contrário, o movimento será classificado apenas pelo tipo de Fluxo Óptico.

5.4 CONSIDERAÇÕES FINAIS

Neste Capítulo buscamos definir as entradas e saídas de cada etapa da metodologia proposta, além de apresentar através de exemplos e ilustrações os resultados intermediários obtidos durante a execução das etapas.

Descrevemos as técnicas de Visão Computacional (cálculo do Fluxo Óptico, histograma de Fluxo Óptico orientado e análise do contorno do objeto no movimento) para gerar as representações numéricas dos movimentos das ações humanas. Os dados numéricos são então apresentados como entrada da etapa de Inteligência Computacional, que irá modelar os dados, para gerar classificadores neurais capazes de determinar o tipo de movimento executado na sequência de imagens.

No próximo Capítulo apresentaremos os resultados obtidos com a implementação de um protótipo da metodologia descrita neste Capítulo. Além disso, descreveremos o conjunto de dados que utilizamos, as técnicas de validação dos classificadores neurais gerados e efetuaremos uma comparação dos resultados que alcançamos com outros encontrados na literatura.

6 RESULTADOS

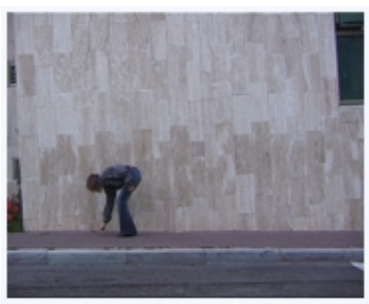
Neste Capítulo apresentamos os resultados obtidos com a implementação de um protótipo da metodologia proposta neste trabalho, descrita no Capítulo 5, cujas técnicas utilizadas foram apresentadas nos Capítulos 2, 3 e 4.

O protótipo foi desenvolvido em duas linguagens. Para o cálculo do Fluxo Óptico utilizamos a biblioteca de Visão Computacional OpenCV. A OpenCV é uma biblioteca de Visão Computacional disponibilizada no site do Projeto Willow Garage [44], escrita em C e em C++ e voltada para os Sistemas Operacionais Linux, Windows e Mac. Foi projetada para prover uma infraestrutura de Visão Computacional de simples utilização para ajudar na construção rápida de aplicações sofisticadas de visão. Esta biblioteca possui mais de 500 funções para visão, tais como, inspeção de fabricação de produtos, imagens médicas, segurança, interface de usuários, calibração de câmera, visão estéreo, e robótica [3]. Para a análise do contorno dos objetos durante o movimento e para a implementação dos algoritmos das redes neurais artificiais, desenvolvemos diversas funções no Matlab. O Matlab é uma linguagem de alto nível e um ambiente de desenvolvimento, que permite a implementação de tarefas computacionais mais rapidamente do que em linguagens tradicionais, tais como C, C++ e Java.

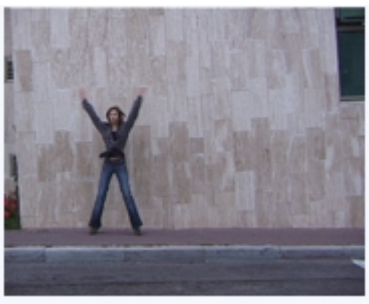
Com o protótipo desenvolvido foi possível investigar a metodologia proposta, identificando suas principais características, suas vantagens e suas desvantagens. O teste da metodologia para aplicações em tempo real não foi nosso foco. Desta forma, neste Capítulo buscamos apresentar o conjunto de dados utilizado para testar a metodologia, descrever os tipos de experimentos que foram realizados, definir a forma de validação dos resultados, bem como analisar os resultados encontrados.

6.1 CONJUNTO DE DADOS UTILIZADO

A fim de que possamos verificar o desempenho da metodologia proposta neste trabalho, utilizamos a base de dados disponibilizada pelo Laboratório de Visão Computacional do Instituto de Ciências Weizmann [12], a qual contém 83 sequências de imagens mostrando nove pessoas diferentes, cada uma delas executando 9 ações naturais: 1 - abaixar e levantar, 2 - fazer polichinelo, 3 - pular para cima e para baixo, 4 - pular para frente, 5 - correr, 6 - andar lateralmente, 7 - andar para frente, 8 - acenar com um braço e 9 - acenar com os dois braços, as quais estão ilustradas na Figura 6.1. As ações *5 - correr* e *7 - andar para frente* possuem uma sequência de imagens a mais que as demais ações. Além das nove ações executadas pelas nove pessoas, uma das pessoas repetiu o movimento destas ações, porém no sentido inverso. Na primeira sequência de imagens a pessoa correu (ou andou) para a direita e na segunda para a esquerda. Todas as sequências de imagens foram gravadas em baixa resolução, com 50 quadros por segundo, de dimensões 180×144 pixels.

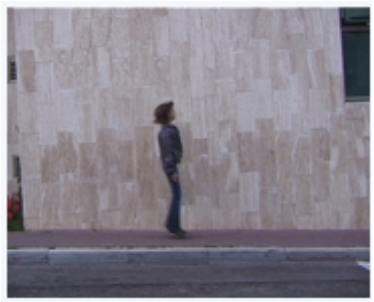


(a) Abaixar e levantar (com 9 sequências de imagens)



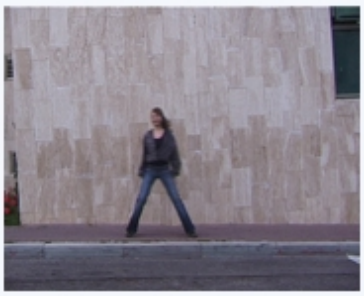
(b) Fazer polichinelo (9)

(c) Pular para cima e para baixo (9)

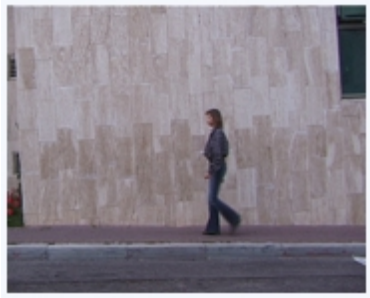


(d) Pular para frente (9)

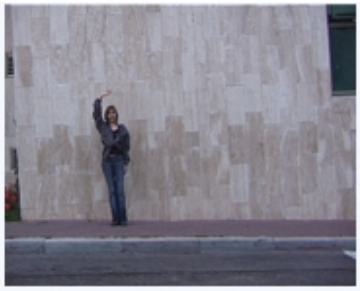
(e) Correr (10)



(f) Andar lateralmente (9)



(g) Andar para frente (10)



(h) Acenar com um braço (9)

(i) Acenar com os dois braços (9)

Figura 6.1: Ações naturais executadas no conjunto de sequências de imagens disponibilizado pelo Laboratório de Visão Computacional do Instituto de Ciências Weizmann [12].

6.2 EXPERIMENTOS

Para justificar a escolha da ordem na qual as técnicas de cálculo de Fluxo Óptico e análise do contorno do objeto durante o movimento serão aplicadas, executamos três tipos de experimentos. Nos experimentos do Tipo 1, cujo processo está ilustrado na Figura 6.2, executamos o cálculo de apenas um tipo de representação do movimento e apresentamos esta representação para o treino e teste das RNAs. Entre estas representações utilizamos o HOOF com 8 *bins*, o HOOF com 16 *bins*, o HOOF com 32 *bins*, o contorno descrito pela Assinatura do Centroide, o contorno descrito pelas coordenadas dos pontos de borda normalizados e o contorno descrito pelos Descritores de Fourier.

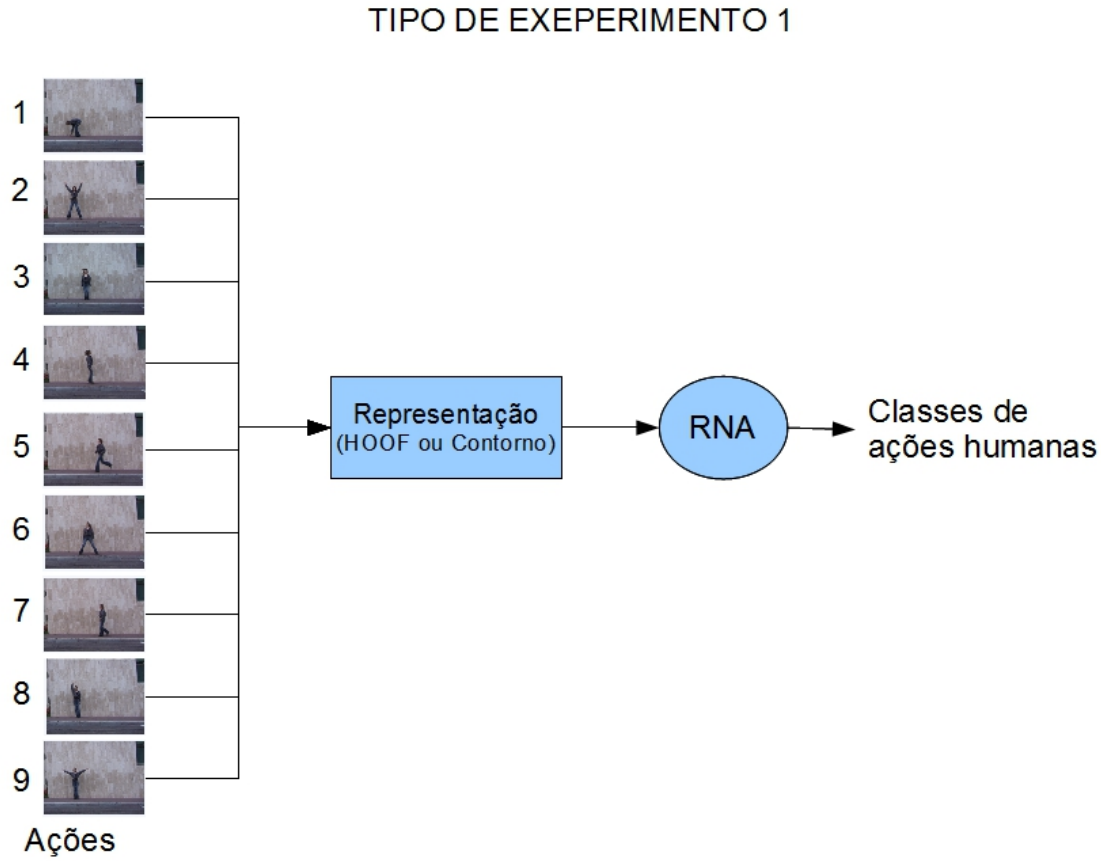


Figura 6.2: Experimento do tipo 1, onde executamos o cálculo de alguma representação do movimento (através do HOOF ou do contorno) e apresentamos esta representação para o treino e teste da RNA.

Nos experimentos do Tipo 2, denominados HOOF+CONTORNO, cujo processo está ilustrado na Figura 6.3, utilizamos os conceitos de comitê de máquinas e geramos três RNA, sendo uma conhecedora de todo o domínio do problema (aqui denominada rede generalista) e outras duas especializadas em parte do domínio (aqui denominadas redes especialistas). Segundo Haykin [8], um comitê de máquinas é um aproximador universal que utiliza o princípio do dividir para conquistar. Ele é capaz de executar uma tarefa computacional complexa, dividindo-a em tarefas computacionais mais simples e combinando suas soluções, com o objetivo de apresentar uma resposta que soluciona a tarefa principal. As tarefas computacionais mais simples são resolvidas por agentes especialistas.

A rede generalista recebe como entrada as ações representadas pelo HOOF (com 8, 16 e 32 *bins*), as quais foram agrupadas heuristicamente de acordo com o perfil de Fluxo Óptico que cada ação gera (Seção 5.3). As ações de teste que foram classificadas pela rede generalista como pertencentes aos tipos de movimento 4 são incluídas no grupo de teste da rede especialista do contorno dos movimentos do tipo 4. O mesmo acontece para os exemplos classificados como do tipo de movimento 5. O grupo de treino da rede especialista 4 (ou 5) é formado pelos exemplos de treino agrupados no tipo de movimento 4 (ou 5), utilizados pela rede generalista. As demais ações são classificadas conforme definido pela rede generalista.

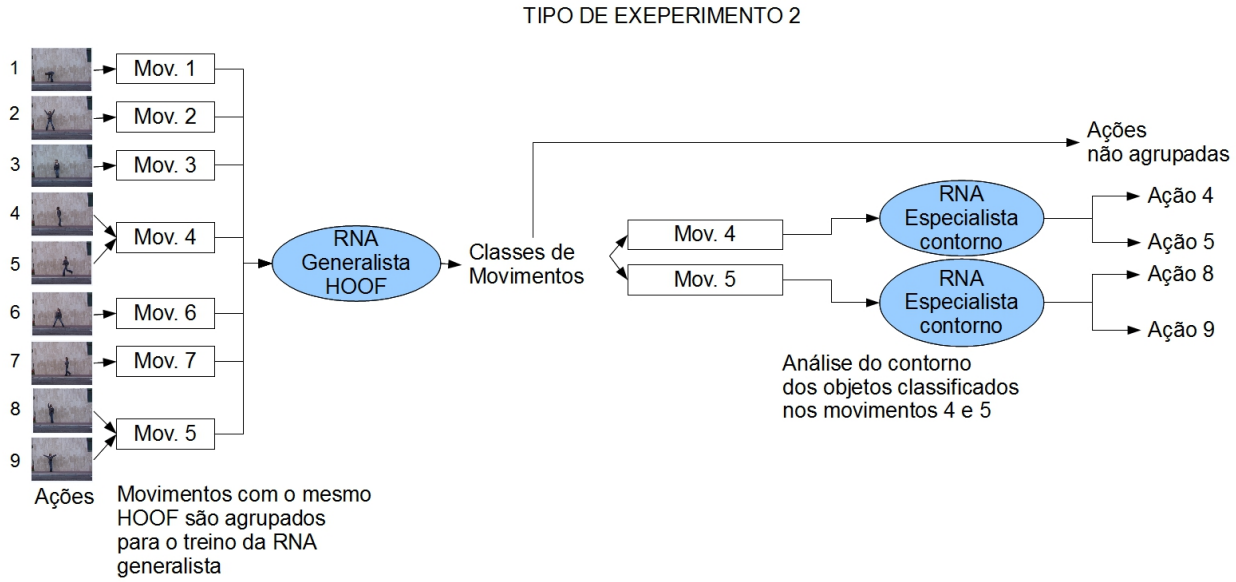


Figura 6.3: Experimento do tipo 2, também denominado HOOF+CONTORNO, onde executamos o agrupamento de ações com o mesmo perfil de Fluxo Óptico e utilizamos uma RNA generalista para classificar por tipo de movimento. Em seguida apresentamos os movimentos que foram agrupados para duas RNA especializadas nas possíveis contornos daqueles movimentos, que irão separar os movimentos agrupados de acordo com as ações originais.

Os experimentos do Tipo 3, também denominados CONTORNO+HOOF, ilustrado na Figura 6.4, utiliza processo semelhante. São utilizadas duas RNA, sendo uma generalista e uma especialista. A rede generalista classifica por tipo de contorno do objeto durante a execução do movimento. A rede especialista separa o que foi agrupado para a rede generalista, de acordo com o HOOF do movimento.

6.3 VALIDAÇÃO DAS REDES NEURAIS GERADAS

Os algoritmos de aprendizagem, como as Redes Neurais Artificiais, necessitam de validação dos resultados alcançados ou do desempenho do aprendizado alcançado. Por meio desta validação podemos estimar a capacidade de generalização da rede neural, isto é, estimar a capacidade da rede apresentar bom desempenho quando dados diferentes daqueles utilizados durante o treinamento forem apresentados a ele. A estimativa da capacidade de generalização é uma tarefa essencial para decidirmos quando as redes poderão ser utilizados com dados reais. Para calculá-la utilizamos um método estatístico conhecido como Valida-

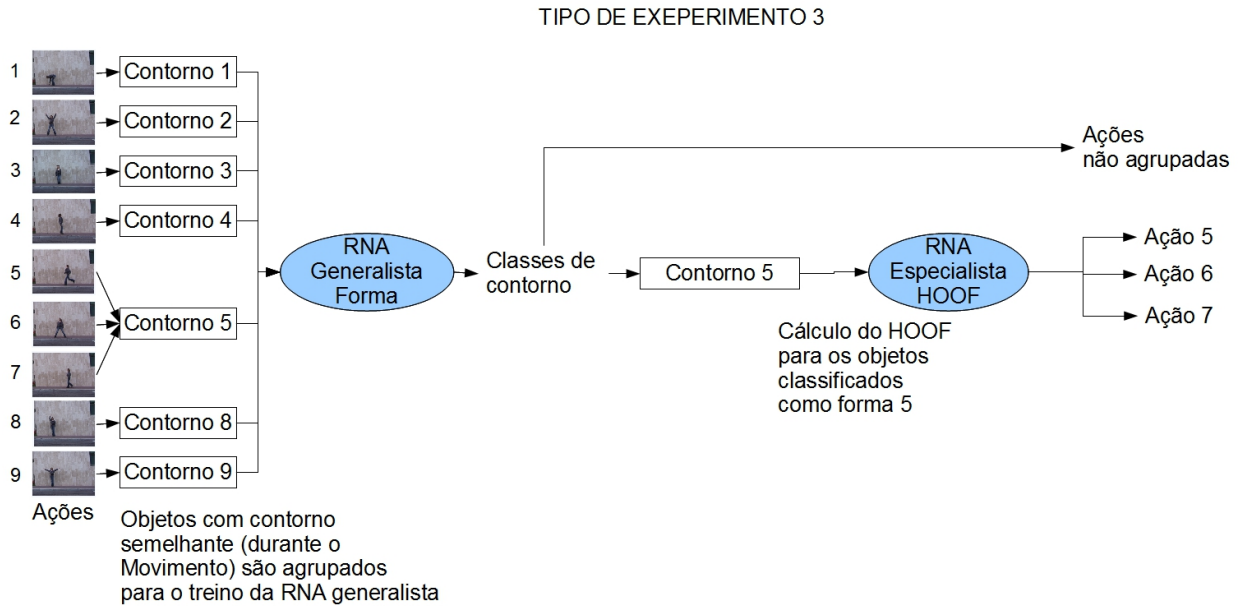


Figura 6.4: Experimento do tipo 3, também denominado CONTORNO+HOOF, onde executamos o agrupamento de ações com o mesmo perfil de contorno e utilizamos uma RNA generalista para classificar por tipo de contorno. Em seguida apresentamos os contornos que foram agrupados para uma RNA especializada nos possíveis perfis de Fluxo Óptico (HOOF) desta contorno, que irá separar o contorno agrupado de acordo com as ações originais.

ção Cruzada. Segundo Refaeilzadeh *et al.* [13], a validação cruzada é um método estatístico de avaliação e comparação de algoritmos de aprendizagem. Neste método os dados disponíveis são divididos em dois grupos: um grupo para o treinamento (ou aprendizado) de uma rede e o outro para validar a rede gerada. Refaeilzadeh *et al.* [13] afirmam que a Validação Cruzada possui dois objetivos principais:

1. Estimar o desempenho de um modelo de aprendizado que foi produzido a partir de um conjunto de dados apresentado a determinado algoritmo. Desta forma, a validação cruzada irá mensurar a capacidade de generalização do modelo.

2. Para comparar o desempenho de dois ou mais algoritmos diferentes, encontrando o melhor dentre eles.

Para alcançar estes dois objetivos, alguns procedimentos de validação foram propostos. Dentre eles podemos citar: a validação por re-substituição; a validação deixe um grupo fora (também conhecida como *hold-out validation*; a validação cruzada *k-fold*; e a validação cruzada deixe um de fora (LOOCV, do inglês *Leave-one-out cross-validation*).

Na validação por re-substituição o algoritmo de treinamento é executado com todos os dados disponíveis. O teste é efetuado com o mesmo conjunto de dados. Trata-se de um método de simples implementação, com baixo custo computacional, que não testa a capacidade de generalização do resultado do treinamento. Pode avaliar como bom o desempenho de um sistema que apresentaria resultados ruins com dados diferentes daqueles utilizados durante o treinamento. Este tipo de validação deve ser utilizada em

domínios de problemas bem definidos, em que é possível utilizar uma base de dados que cubra todo o problema. Também é utilizado para atestar a eficácia de um método, ou seja, se ele não consegue classificar corretamente os dados para os quais foi treinado, não conseguirá bons resultados com dados desconhecidos, podendo ser descartado.

No método de validação deixe um grupo fora os dados pertencentes ao grupo de treino são diferentes dos dados do grupo de teste. Este método possui resultado altamente dependente da escolha feita durante a separação dos dados disponíveis entre os grupos de treino e teste. É comum utilizar entre 50% e 70% dos dados no grupo de treino e o restante no grupo de teste.

Na validação cruzada k-*fold* os dados disponíveis são particionados em k grupos mutuamente exclusivos de tamanhos iguais, denominados $D_1, D_2, D_3, ..., D_k$. O treinamento e o teste são executados k vezes. Na iteração i o grupo D_i é definido como grupo de teste e os demais $k-1$ grupos são agrupados para formar o conjunto de treino. Desta forma, na primeira iteração os conjuntos $D_2, D_3, ..., D_k$ serão utilizados agrupadamente com o conjunto de treino, para obter o primeiro modelo, que será testado pelo grupo D_1. Semelhantemente, na segunda iteração os conjuntos $D_1, D_3, D_4, ..., D_k$ são utilizados para o treino e geração do modelo e o conjunto D_2 irá testar este modelo gerado. Refaeilzadeh *et al.* [13] ilustram a validação cruzada 3-*fold*, conforme apresentado na Figura 6.5. O erro total deste método é obtido pela soma dos erros de todas as execuções. A vantagem em relação aos métodos apresentados anteriormente está no fato de que cada dado é utilizado no treinamento $k-1$ vezes e é utilizado no teste uma vez. É um método que apresenta uma boa estimativa para a capacidade de generalização do modelo gerado. É comum utilizar este método com $k = 10$.

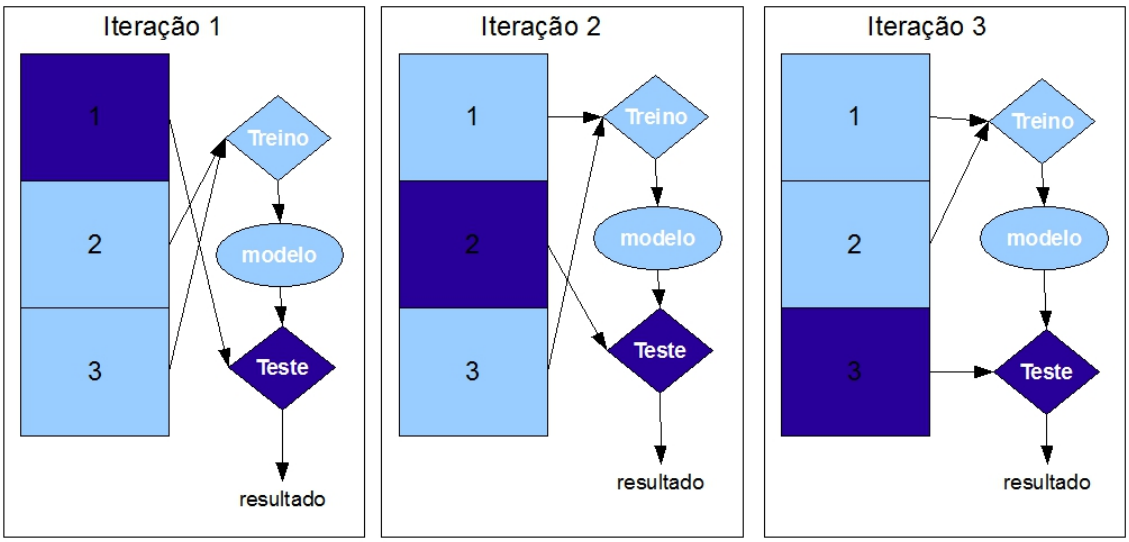


Figura 6.5: Procedimento da validação cruzada 3-*fold*. Adaptado de [13].

A validação cruzada deixa um fora é um caso especial da validação cruzada k-*fold*, onde $k = n$, sendo n a quantidade de dados no conjunto original. Neste método cada conjunto de teste possui apenas um registro. Sua principal vantagem está no fato de ser capaz de utilizar tantos dados quanto possíveis

para treinamento do algoritmo. Por outro lado, esta abordagem possui alto custo computacional, já que o procedimento de treino e teste é executado n vezes. É comumente utilizado quando a quantidade de dados disponíveis para a execução do algoritmo é pequena. Cada um destes métodos possui suas vantagens e desvantagens. Refaeilzadeh *et al.* [13] apresentam uma comparação entre eles.

6.4 RESULTADOS EXPERIMENTAIS

Nesta Seção apresentamos os resultados dos experimentos realizados com o protótipo da solução descrita neste trabalho. A Tabela 6.1 demonstra os resultados alcançados com os experimentos do Tipo 1. Nestes experimentos utilizamos as técnicas de HOOF e da análise do contorno do objeto durante o movimento como maneira de representação dos movimentos ocorridos nas sequências de imagens. Os padrões gerados por estas técnicas foram apresentados à Redes Neurais Artificiais SOM e LVQ, de tal forma que implementavam a classificação adaptativa de padrões. O desempenho das redes geradas foi validado com as técnicas de validação por re-substituição (RV), validação cruzada 10-*fold* (10FOLD) e validação cruzada deixe um fora (LOOCV). Os parâmetros utilizados por estes algoritmos estão descritos nas Seções do Capítulo 5.

Para os experimentos do Tipo 1 notamos que considerando a técnica de validação por re-substituição as representações do movimento pelos HOOFs apresentaram os melhores resultados, sendo que quando utilizamos 16 *bins* no histograma do Fluxo Óptico o resultado alcançou 97.59%. Como sabemos esta técnica de validação não é boa para mensurar a capacidade de generalização do algoritmo. Quando utilizamos as técnicas de Validação Cruzada 10FOLD e LOOCV notamos que os resultados não são tão bons. Isso indica que com estas representações de padrões as redes geradas tornam-se dependentes dos dados utilizados para o treinamento. A depender do grau de criticidade da aplicação onde a proposta será inserida, as redes geradas podem ser considedaras inadequadas. Em geral, quando utilizamos apenas a representação do movimento por HOOF alcançamos melhores resultados do que quando utilizamos as representações dos contornos dos objetos durante os movimentos através das assinaturas do centroide, das bordas normalizadas e dos descritores de fourier.

Tipo Validação	**HOOF 8 *bins***	**HOOF 16 *bins***	**HOOF 32 *bins***	**Assinatura Centroide**	**Borda Normal**	**Descritores de Fourier**
Re-substituição	91.57%	97.59%	95.18%	81.93%	86.75%	85.54%
Leave-one-out cross validation	63.86%	75.90%	80.72%	62.65%	71.08%	53.01%
10-Fold cross validation	73.49%	77.11%	85.54%	62.65%	73.49%	56.63%

Tabela 6.1: Resultados dos experimentos do Tipo 1.

Nos experimentos do Tipo 2 unimos duas técnicas para representar o movimento. Na primeira etapa executamos as classificações dos movimentos mediante avaliação dos padrões de HOOFs com 8,16 e 32 *bins*. Em seguida, utilizamos os descritores de contorno do objeto durante o movimento para executar a classificação especializada dos grupos de movimentos que foram identificados na primeira etapa como pertencentes ao mesmo tipo de Fluxo Óptico, como descrito na Seção 6.2. Como anteriormente, validamos as redes geradas com as três técnicas de validação já apresentadas. Os resultados estão apresentados na

Tabela 6.2. Destacamos que utilizando HOOF com 32 *bins* juntamente com a representação de contorno por Bordas Normalizadas com 100 pontos, alcançamos 93.98% de acerto, o qual foi estimado utilizando a validação LOOCV. Com o mesmo tipo de validação e com o mesmo número de *bins*, porém utilizando a representação de contorno pelos Descritores de Fourier, alcançamos 91.57%. Na estimativa pelo método de validação 10-fold alcançamos bons resultados para as representações por Assinatura do Centroide e por Borda Normalizada, obtendo 91.57% e 92.77%. Considerando 16 *bins* alcançamos bons resultados com a representação de contorno pela Assinatura do Centroide e a validação LOOCV (91.57%) e com a representação por Bordas Normalizadas e validação 10-Fold (92.77%).

Número *bins*	**Tipo Validação**	**HOOF + Assinatura Centroide**	**HOOF + Borda Normal**	**HOOF + Descritores Fourier**
8	***Re-substituição***	90.36%	90.36%	95.18%
	Leave-one-out cross validation	74.70%	78.31%	79.52%
	10-Fold cross validation	74.70%	79.52%	77.11%
16	***Re-substituição***	97.59%	97.59%	97.59%
	Leave-one-out cross validation	**91.57**%	89.16%	86.75%
	10-Fold cross validation	90.36%	**92.77**%	89.16%
32	***Re-substituição***	98.80%	98.80%	100%
	Leave-one-out cross validation	89.16%	**93.98%**	**91.57%**
	10-Fold cross validation	**91.57**%	**92.77**%	89.16%

Tabela 6.2: Resultados dos experimentos do Tipo 2.

A fim de avaliar quais seriam os resultados caso a ordem de aplicação das técnicas fossem invertidas executamos os experimentos do Tipo 3, que seguiram o mesmo perfil dos experimentos do Tipo 2, com o número de *bins* variando entre 8, 16 e 32 e com os resultados de classificação validados pelas técnicas de re-substituição, 10-*fold* e validação cruzada deixe um de fora. Os resultados alcançados, apresentados na Tabela 6.3, são inferiores aos encontrados com os experimentos do Tipo 2. A melhora dos resultados, quando comparados com aqueles obtidos nos experimentos do Tipo 1, está variando entre 10% e 15%.

Notamos que os experimentos do Tipo 2 foram aqueles que apresentaram os melhores resultados, alcançando até 93.98% de acerto. Com o objetivo de tentar identificar como os resultados são afetados a medida que o número de *bins* é alterado, utilizando as técnicas na ordem HOOF + CONTORNO (Tipo 2), variamos o número de *bins* dos HOOFs entre 5 e 100, validando as redes geradas com a técnica LOOCV. Os resultados alcançados estão apresentados na Tabela 6.4.

Percebemos que a maior parte dos resultados encontrados apresentam taxa de erro inferior a 10%.

Notamos apenas um comportamento de tendência de melhora dos resultados derivados do aumento do número de *bins*, encontrado no intervalo entre 5 e 15 *bins*. No restante não encontramos outras tendências de melhora ou piora dos resultados que possam ser associadas ao aumento ou redução do número de *bins*. Com estes resultados, encontramos dois mínimos locais, com o primeiro ocorrendo em torno de 15 *bins* e com o segundo ocorrendo em torno de 60 e 65 *bins*.

Os melhores resultados foram obtidos quando utilizamos 15 e 60 bins para as técnicas HOOF + Borda

Número *bins*	Tipo Validação	Assinatura Centroide + HOOF	Borda Normal + HOOF	Descritores Fourier + HOOF
8	***Re-substituição***	89.16%	89.16%	84.34%
	Leave-one-out cross validation	67.47%	69.88%	51.81%
	10-Fold cross validation	66.27%	67.47%	48.19%
16	***Re-substituição***	93.98%	93.98%	87.95%
	Leave-one-out cross validation	79.52%	79.52%	66.27%
	10-Fold cross validation	74.70%	75.90%	53.01%
32	***Re-substituição***	96.39%	96.39%	87.95%
	Leave-one-out cross validation	75.90%	80.72%	65.06%
	10-Fold cross validation	79.52%	78.31%	66.27%

Tabela 6.3: Resultados dos experimentos do Tipo 3.

Número *bins*	HOOF + Borda Normal	HOOF + Descritores Fourier
5	83.13%	84.34%
10	87.95%	80.72%
15	**96.39%**	**95.18%**
20	90.36%	93.98%
25	87.95%	89.16%
30	92.77%	91.57%
35	90.36%	91.57%
40	91.57%	92.77%
45	87.95%	93.98%
50	91.57%	93.98%
55	90.36%	91.57%
60	**95.18%**	92.77%
65	93.98%	**95.18%**
70	85.54%	90.36%
75	92.77%	89.16%
80	91.57%	91.57%
85	89.16%	89.16%
90	87.95%	91.57%
95	92.77%	92.77%
100	92.77%	87.95%

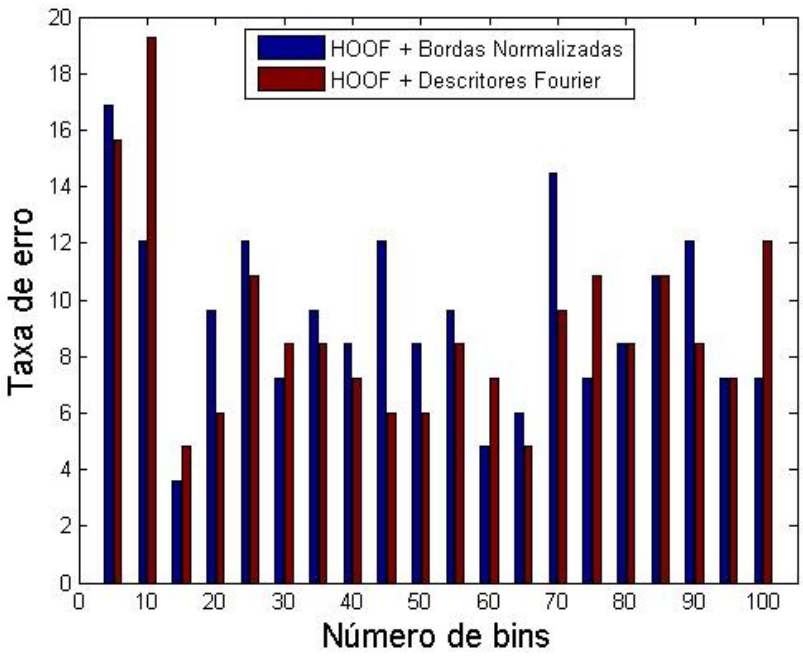


Tabela 6.4: Resultados dos experimentos do Tipo 2, considerando as técnicas HOOF + Bordas Normalizadas e HOOF + Descritores de Fourier, com o número de *bins* variando de 5 a 100 e o número de pontos nas bordas normalizadas igual a 100, e validados com a técnica LOOCV.

Normalizada, onde alcançamos 96.39% e 95.18% de acerto, respectivamente. Semelhantemente, utilizando 15 e 65 bins para as técnicas HOOF + Descritores de Fourier alcançamos 95.18% de acerto. Nas Tabelas 6.5, 6.6, 6.7 e 6.8 apresentamos as matrizes de confusão destes experimentos, que demonstram grande semelhança nos erros de classificação.

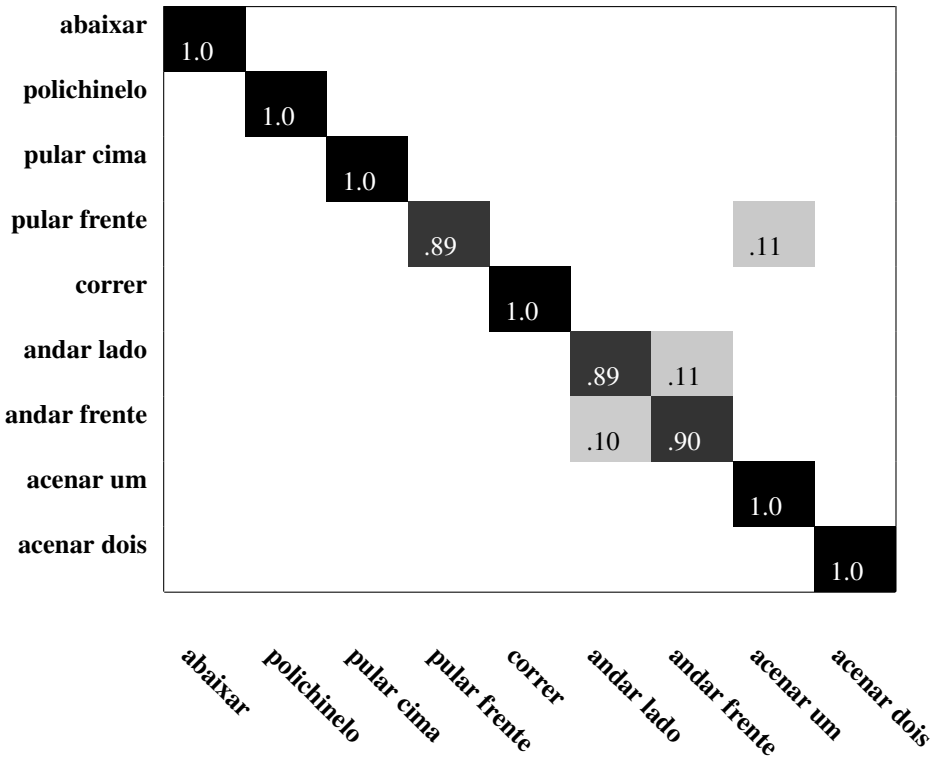


Tabela 6.5: Matriz de Confusão para os experimentos do Tipo 2, HOOF + Borda Normalizada, com histogramas de 15 bins, com 96.39% de acerto.

Com o objetivo de comparar os resultados encontrados com a implementação do protótipo da metodologia apresentada neste trabalho, sumarizamos na Tabela 6.9 os trabalhos relacionados descritos na Seção 1.1, os quais utilizaram algoritmos de Visão e Inteligência Computacional para reconhecimento do movimento executado nas sequências de imagens e as técnicas de Validação Cruzada, descritas na Seção 6.3, para estimar a capacidade de generalização das técnicas de reconhecimento.

6.5 CONSIDERAÇÕES FINAIS

Neste Capítulo realizamos experimentos com o objetivo de validar a metodologia que propomos. Utilizamos um conjunto de dados comumente utilizado para validar métodos que tratam problemas semelhantes. Apresentamos os tipos de experimentos que realizamos, alterando a ordem de utilização dos tipos de representações e a quantidade de classificadores neurais envolvidos no processo. Também descrevemos os métodos de validação de algoritmos de aprendizado que podem ser aplicados com o objetivo de estimar a capacidade de generalização dos classificadores neurais.

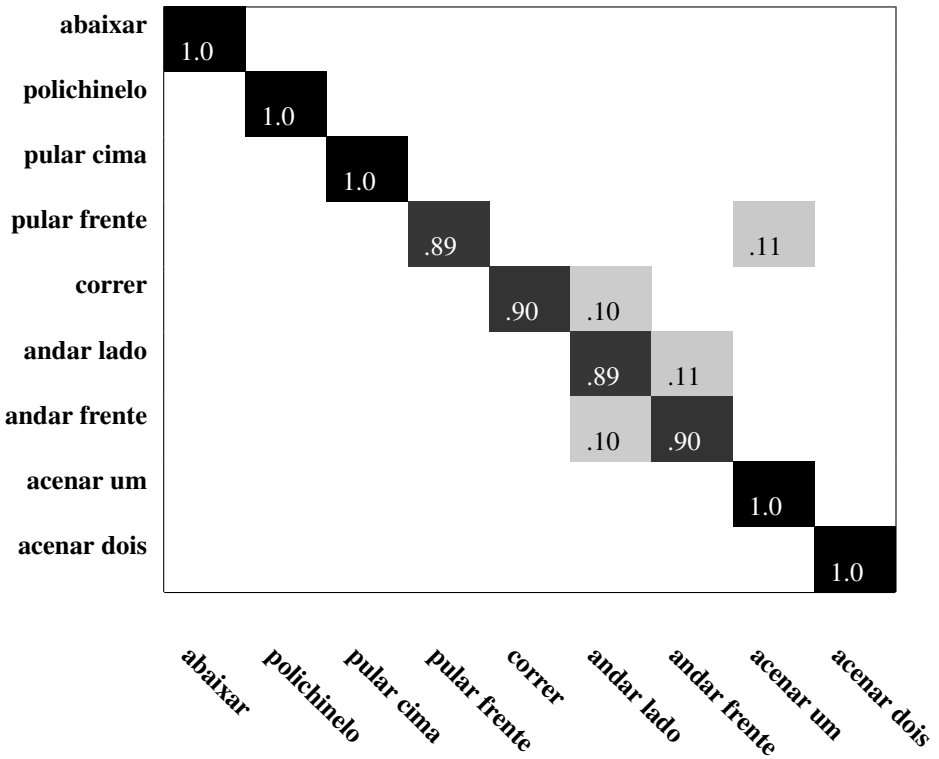


Tabela 6.6: Matriz de Confusão para os experimentos do Tipo 2, HOOF + Borda Normalizada, com histogramas de 60 bins, com 95.18%.

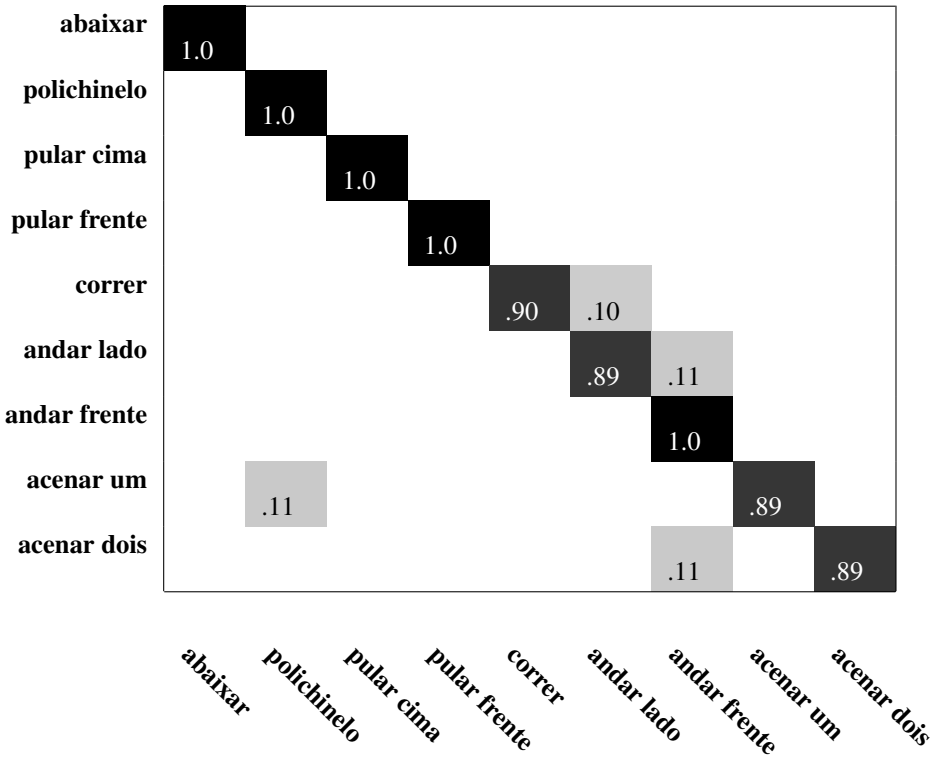


Tabela 6.7: Matriz de Confusão para os experimentos do Tipo 2, HOOF + Descritores Fourier, com histogramas de 15 bins, com 95.18%.

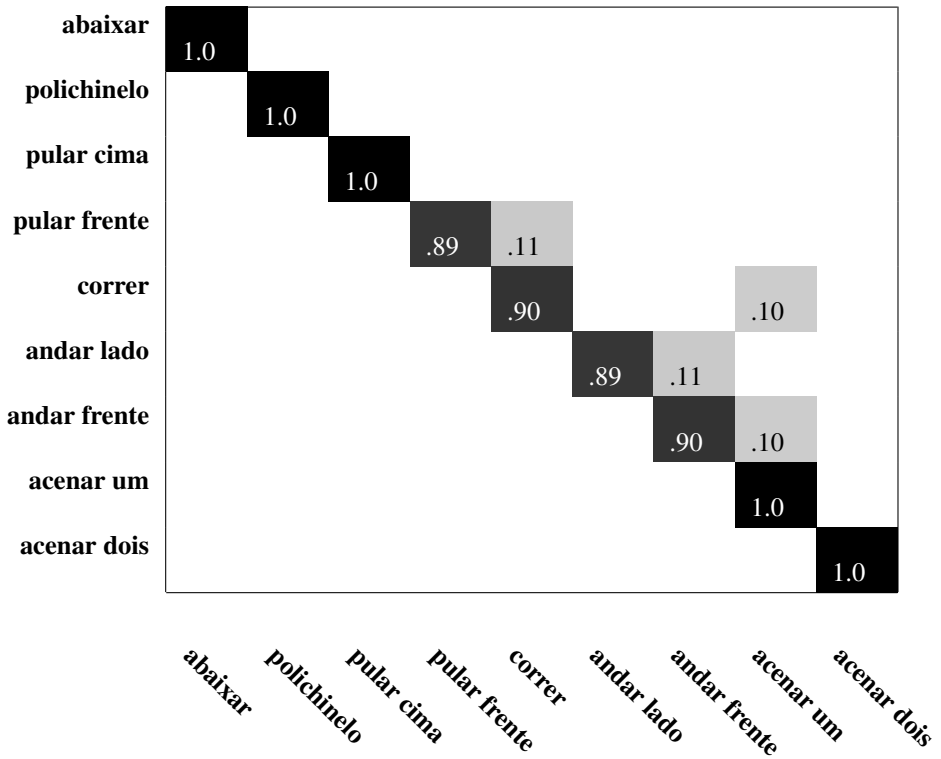


Tabela 6.8: Matriz de Confusão para os experimentos do Tipo 2, HOOF + Descritores Fourier, com histogramas de 65 bins, com 95.18%.

Método	**Técnica de Visão**	**Classificador**	**Validação**	**Acerto (%)**
Niebles e Li [20]	Formas e Pontos de Interesse	SVM	LOOCV	72.8
Ali *et al.*[19]	Trajetória	Vizinho Mais Próximo	LOOCV	92.6
Wang, Huang e Tan [15]	Fluxo Óptico	AdaBoost	LOOCV	93.3
Thurau e Hlavac [23]	Gradientes Orientados	Vizinho Mais Próximo	LOOCV	94.4
Chaudhry *et al.* [4]	HOOF	Vizinho Mais Próximo	LOOVC	94.4
Nosso método	**HOOF e Análise de Contorno**	**RNA (SOM+LVQ)**	**LOOCV**	**96.4**
Wang e Leckie [18]	Silhueta	*k-means*	LOOCV	96.8
Jhuang *et al.*[21]	Movimentos e Formas	SVM	Hold-Out	98.8

Tabela 6.9: Comparação do desempenho de classificação de diversos propostas que utilizaram a base de dados de Weizmann.

Por meio dos resultados encontrados notamos que nossa metodologia é competitiva quando comparada aos demais trabalhos já publicados. Resolver problemas do mundo alcançando 96.4% de acurária não é uma tarefa trivial.

No próximo Capítulo encerramos esta dissertação com as considerações finais do autor, as principais contribuições deste trabalho de pesquisa e as sugestões de trabalhos futuros.

7 CONCLUSÕES

Conforme apresentado no Capítulo 1, a análise automatizada de sequências de imagens pode auxiliar diversas áreas da sociedade na execução de tarefas tidas como cotidianas e que consomem tempo, dinheiro e outros recursos. Tais recursos poderiam ser aplicados em outras tarefas, para gerar (ou acrescentar) melhores (ou outros) resultados. Além disso, esta automatização pode otimizar os recursos dispensados na análise de sequências de imagens. Este trabalho teve como principal objetivo o desenvolvimento de uma metodologia capaz de executar a análise e o reconhecimento automatizado de ações humanas em sequências de imagens. Com base no estudo de diversos trabalhos que propuseram métodos com objetivos semelhantes (Seção 1.1), estabelecemos que a nossa metodologia deveria: processar as sequências de imagens que continham a ação humana, gerar padrões de representação das ações humanas e a implementar algoritmos de reconhecimento de padrões para reconhecer a ação humana.

Para atender tais requisitos, a metodologia foi desenvolvida combinando técnicas da Visão Computacional e da Inteligência Computacional (Figura 5.1). Por meio das técnicas de visão implementamos métodos de processamento das sequências de imagens e de representação das ações humanas executadas. Com as técnicas de Inteligência Computacional executamos o reconhecimento dos padrões gerados na etapa de Visão Computacional e, por consequência, das ações humanas executadas.

Nos Capítulos 2 e 3 descrevemos as técnicas de Visão Computacional utilizadas na metodologia. A primeira técnica apresentada é a estimativa do cálculo do Fluxo Óptico (Seção 2.2), uma metodologia utilizada para obtenção de informações da ação humana executada na sequência de imagens. Por meio dela, são estimados vetores de Fluxo Óptico que representam o deslocamento ocorrido nas imagens. Como a quantidade de vetores gerados é grande, também utilizamos a técnica dos Histogramas Orientados de Fluxo Óptico (Seção 2.3) para estimar uma distribuição destes vetores. Como os vetores de Fluxo Óptico e sua respectiva distribuição por histogramas podem possuir padrões semelhantes para ações distintas, utilizamos as técnicas de análise do contorno do objeto para separar estes padrões. Entre estas técnicas destacamos a aplicação das tarefas de detecção do contorno dos objetos (Seção 3.4.1) e de geração de representações para estes contornos (Seções 3.5, 3.6 e 3.7).

Os padrões gerados pelos Histogramas Orientados de Fluxo Óptico, Coordenadas de Bordas Normalizadas, Assinaturas da Distância do Centroide e Assinaturas dos Descritores de Fourier, que são representantes das ações humanas executadas nas sequências de imagens, foram apresentados no treinamento das RNA SOM e LVQ (Seções 4.1.1 e 4.1.2) para que pudéssemos investigar a robustez e eficácia de cada tipo de representação, mediante a avaliação do poder de generalização dos modelos neurais resultantes do treinamento.

Por meio dos resultados, apresentados na Seção 6.4, percebemos que o método foi capaz de alcançar taxas de erros de classificação aceitáveis e compatíveis com aqueles publicados na literatura. Estes resultados demonstram a robustez da técnica perante uma base de dados que possui diferentes tipos de ações humanas. Os testes de variação na quantidade de bins demonstraram que não é necessário aumentar esta quantidade para obtermos resultados melhores. Com 15, 60 e 65 bins a técnica se mostrou efetiva. Tais re-

sultados foram obtidos sem a necessidade de etapas de pré-processamento da imagem, como por exemplo a localização e extração da região que contém a pessoa nos quadros.

Podemos atribuir o sucesso do método proposto à integração de diferentes técnicas, com o objetivo principal de resolver um problema complexo.

Nas próximas Seções apresentamos algumas vantagens do nosso método quando o comparamos às técnicas descritas na Seção 1.1 e em seguida algumas sugestões de trabalhos futuros.

7.1 COMPARAÇÃO COM OUTROS MÉTODOS

Na Tabela 6.9 apresentamos uma comparação dos resultados alcançados com a implementação do nosso método com aqueles apresentados por outros autores. Percebemos que no método proposto por Wang, Huang e Tan [15], que alcançou 93.3% de acerto, os autores também utilizaram representação por histograma de Fluxo Óptico, a qual foi complementada com informações estatísticas do Fluxo Óptico e do contorno e trajetória do objeto durante o movimento. A vantagem do nosso método com relação ao proposto por Wang, Huang e Tan [15] está principalmente na minimização do esforço computacional necessário para a geração do classificador. Enquanto nós utilizamos apenas três classificadores neurais, eles utilizaram um classificador gerado através do algoritmo Adaboost Multi-classe, que exige o treinamento de diversos classificadores fracos, os quais reunidos produzem um resultado final melhorado.

Já o método proposto por Thurau and Hlavac [23], que alcançou 94.4% de acerto, necessita executar um processo de classificação prévio na sequência de imagens para localizar a região que contém uma pessoa. Viola e Jones [45] apresentam uma técnica que pode realizar esta classificação prévia. Além do resultado ligeiramente melhor, uma das vantagens do nosso método com relação ao proposto por Thurau and Hlavac [23] está na inexistência de etapas de classificação prévia de partes das imagens que compõe o vídeo.

Chaudhry *et al.* [4] também alcançaram resultado ligeiramente inferior ao que encontramos. Assim como a deles, nossa proposta não necessita de qualquer pré-processamento ou detecção humana. Com a inclusão do passo de análise do contorno do objeto, conseguimos melhorar os resultados encontrados, especialmente para os casos onde ações distintas eram representadas por HOOF semelhantes.

Wang e Leckie [18] propuseram uma metodologia que alcançou 96.8% de acerto, resultado bem próximo ao que alcançamos. Houveram alguns erros ao tentar diferenciar as ações das classes one-hand wave e two-hands wave. Seguindo nosso método estas classes são agrupadas na etapa de análise do perfil do Fluxo Óptico e em seguida geramos uma RNA especialista capaz diferenciá-las utilizando a representação por coordenadas de borda normalizada, obtendo 100% de acerto.

7.2 TRABALHOS FUTUROS

Sabemos que os resultados apresentados refletem o comportamento da técnica perante a base de dados utilizada. Trabalhos posteriores a este poderiam testar esta mesma técnica perante outras bases de dados,

nas quais os sujeitos executam ações diferentes daquelas aqui testadas e também onde ocorra mudança no *background* dos quadros das sequências de imagens. Isso reforçaria a robustez e eficácia do método.

Também poderíamos incluir análise e testes estatísticos nos dados gerados, a fim de identificar possíveis características que pudessem ser utilizadas para classificação.

Outra característica que serviria para evoluir a metodologia seria adaptá-la a aplicações que exijam processamento e resposta em tempo real. Assim poderíamos identificar pontos de gargalo no método, que deveriam ser modificados, com o objetivo principal de alcançarmos o menor tempo de resposta possível.

A classificação de sequências de imagens em domínios de problemas que não envolvam serem humanos executando movimentos também poderia ser testada. Como exemplo poderíamos citar a identificação da execução de movimentos de trânsito não permitidos (conversão indevida ou mudança de faixa não permitida, por exemplo). A depender do ângulo em que as imagens serão capturadas não adiantaria avaliarmos o contorno do veículo, já que ele poderia ter contorno fixo. A técnica deveria ser avaliada, e talvez adaptada, para atender necessidades semelhantes a esta.

Outra demanda seria a classificação de múltiplas ações humanas no mesmo quadro da imagem. Deveríamos incluir um passo para segmentar e localizar as pessoas no quadro e dai aplicar nosso método para classificar as ações individuais.

As ideias apresentadas neste trabalho servem como ponto de partida para o desenvolvimento de uma técnica que possui campo de aplicação mais abrangente, contribuindo para o desenvolvimento de algoritmos de aprendizado de máquina voltados para Visão Computacional. Com nosso trabalho demonstramos que a integração destas técnicas pode prover métodos capazes de resolver problemas do mundo real, com taxas de erros aceitáveis, servindo como motivação e inspiração para outros pesquisadores destas áreas.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] MINETTO, R. *Detecção robusta de movimento de camera em videos por analise de fluxo otico ponderado*. Dissertação (Dissertação de Mestrado) — Instituto de Computação, Universidade Estadual de Campinas, Campinas, SP, Brasil, 2007.

[2] BEAUCHEMIN, S. S.; BARRON, J. L. The computation of optical flow. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 27, p. 433–466, September 1995.

[3] BRADSKI, G.; KAEHLER, A. *Learning OpenCV*. 1. ed. Gravenstein Highway North, Sebastopol, CA, USA: O'Reilly Media, 2008. ISBN 978-0-596-51613-0.

[4] CHAUDHRY, R. et al. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. [S.l.: s.n.], 2009. p. 1932 –1939.

[5] GHUNEIN, A. *Contour Tracing*. june 2011. Disponível em: <http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/index.html>.

[6] COSTA, L.; CESAR, R. *Shape Analysis and Classification: Theory and Practice*. Boca Raton, FL, USA: CRC Press, Inc., 2000.

[7] COSTA, J. A. F. *Classificação Automática e Análise de Dados por redes neurais auto-organizáveis*. Tese (Tese de Doutorado) — Faculdade de Engenharia Elétrica e de Computação, da Universidade Estadual de Campinas, Campinas, SP, Brasil, Dezembro 1999.

[8] HAYKIN, S. *Neural Networks: a comprehensive foundation*. 2. ed. [S.l.]: Prentice Hall, 1999. ISBN 0132733501.

[9] VESANTO, J. *Using SOM in Data Mining*. Dissertação (Licentiate's thesis) — Department of Computer Science and Engineering. Helsinki University of Technology, April 2000.

[10] TOMASI, C.; KANADE, T. *Detection and Tracking of Point Features*. [S.l.], 1991.

[11] SHI, J.; TOMASI, C. Good features to track. In: *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*. [S.l.: s.n.], 1994. p. 593 – 600.

[12] GORELICK, L. et al. Actions as space-time shapes. In: *The Tenth IEEE International Conference on Computer Vision (ICCV'05)*. [S.l.: s.n.], 2005. p. 1395–1402.

[13] REFAEILZADEH, P.; TANG, L.; LIU, H. *Cross-Validation*. Springer, august 2011. Disponível em: <http://www.public.asu.edu/ ltang9/papers/ency-cross-validation.pdf>.

[14] YILMAZ, A.; JAVED, O.; SHAH, M. Object tracking: A survey. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 38, December 2006.

[15] WANG, S.; HUANG, K.; TAN, T. A compact optical flowbased motion representation for real-time action recognition in surveillance scenes. In: *Image Processing (ICIP), 2009 16th IEEE International Conference on*. [S.l.: s.n.], 2009. p. 1121 –1124.

[16] LUCAS, B. *Generalized Image Matching by the Method of Differences*. Tese (Doutorado) — Robotics Institute, Carnegie Mellon University, July 1984.

[17] LUCAS, B. D.; KANADE, T. An iterative image registration technique with an application to stereo vision. In: *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981. p. 674–679.

[18] WANG, L.; LECKIE, C. Encoding actions via quantized vocabulary of averaged silhouettes. In: *Pattern Recognition (ICPR), 2010 20th International Conference on*. [S.l.: s.n.], 2010. p. 3657 –3660.

[19] ALI, S.; BASHARAT, A.; SHAH, M. Chaotic invariants for human action recognition. In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. [S.l.: s.n.], 2007. p. 1 –8.

[20] NIEBLES, J.; LI, F. A hierarchical model of shape and appearance for human action classification. In: *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*. [S.l.: s.n.], 2007. p. 1 –8.

[21] JHUANG, H. et al. A biologically inspired system for action recognition. In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. [S.l.: s.n.], 2007. p. 1 –8.

[22] MUTCH, J.; LOWE, D. G. Object class recognition and localization using sparse features with limited receptive fields. *Int. J. Comput. Vision*, Kluwer Academic Publishers, Hingham, MA, USA, v. 80, p. 45–57, October 2008. ISSN 0920-5691.

[23] THURAU, C.; HLAVAC, V. Pose primitive based human action recognition in videos or still images. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. [S.l.: s.n.], 2008. p. 1 –8.

[24] HORN, B. P.; SCHUNCK, B. Determining optical flow. *ARTIFICAL INTELLIGENCE*, v. 17, p. 185–203, 1981.

[25] GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. 3nd. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2009.

[26] BARRON, J. L.; FLEET, D. J.; BEAUCHEMIN, S. S. Performance of optical flow techniques. *International Journal of Computer Vision*, Springer Netherlands, v. 12, p. 43–77, 1994.

[27] GALVIN, B. et al. Recovering motion fields: An evaluation of eight optical flow algorithms. In: *British Machine Vision Conference*. [S.l.: s.n.], 1998. p. 195–204.

[28] MCCANE, B. et al. On benchmarking optical flow. *Comput. Vis. Image Underst.*, Elsevier Science Inc., New York, NY, USA, v. 84, p. 126–143, October 2001. ISSN 1077-3142.

[29] BOUGUET, J. *Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the algorithm*. 2002.

[30] ZHANG, D.; LU, G. A comparative study of fourier descriptors for shape representation and retrieval. In: *Proc. of 5th Asian Conference on Computer Vision (ACCV)*. [S.l.]: Springer, 2002. p. 646–651.

[31] TOUSSAINT, G. *Grids, connectivity and contour tracing*. june 2011.

[32] PAVLIDIS, T. *Algorithms for Graphics and Image Processing*. Rockville, Maryland: Computer Science Press, 1982.

[33] PRADHAN, R. et al. Contour line tracing algorithm for digital topographic maps. In: *International Journal of Image Processing (IJIP)*. [S.l.: s.n.], 2010. v. 4, p. 156 – 163.

[34] LUGER, G. *Artificial Intelligence, Structures and Strategies for Complex Problem Solving*. 5. ed. [S.l.]: Addison-Wesley, 2005. ISBN 0321263189.

[35] FAUSETT, L. *Fundamentals of Neural Networks: Architectures, Algorithms And Applications*. Us. [S.l.]: Prentice Hall, 1994. ISBN 0133341860.

[36] MATSUBARA, E. *O Algoritmo de Aprendizado Semi-Supervisionado Co-Training e sua Aplicação na Rotulação de Documentos*. Dissertação (Dissertação de Mestrado) — Instituto de Ciências, Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, Brasil, Maio 2004.

[37] ELMASRI, R.; NAVATHE, S. *Sistemas de Banco de Dados*. 4. ed. [S.l.]: Addison-Wesley, 2005. ISBN 8588639173.

[38] WANG, W. et al. The global fuzzy c-means clustering algorithm. In: *The Sixth World Congress on Intelligent Control and Automation*. [S.l.: s.n.], 2006. v. 1, p. 3604–3607. ISBN 1-4244-0332-4.

[39] KOHONEN, T. The self-organizing map. *Proceedings of the IEEE*, v. 78, September 1990.

[40] HAN, Y. et al. The diagnostic reasoning based on fuzzy self-organizing neural network and its application. *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, v. 4, p. 2453–2457, Aug 2004.

[41] BEZDEK, J. *Fuzzy models and algorithms for pattern recognition and image processing*. [S.l.]: Kluwer Academic Publisher Group, 1999.

[42] BOSCARIOLI, C. *Análise de Agrupamentos baseada na Topologia dos Dados e em Mapas Auto-organizáveis*. Tese (Tese de Doutorado) — Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, 2008.

[43] PERES, S. *Dimensão topológica e mapas auto organizáveis de Kohonen*. Tese (Tese de Doutorado) — Faculdade de Engenharia Elétrica e de Computação, da Universidade Estadual de Campinas, Campinas, SP, Brasil, Setembro 2006.

[44] PROJECT, W. G. *OpenCV - Willow Garage*. Novembro 2011. Http://www.willowgarage.com/pages/software/opencv.

[45] VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. *Proc. CVPR*, v. 1, p. 511–518, 2001.