HOLOS

ISSN: 1518-1634

holos@ifrn.edu.br

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

Brasil

Pereira Florentino, Gustavo Henrique; de Medeiros Valentim, Ricardo Alexsandro; Uchoa Bezerra, Heitor; Gomes de Araújo, Bruno; Teixeira Lacerda, João Marcos; Clemente, Alexandre

TEMPLATE MATCHING COMO TÉCNICA COMPUTACIONAL APLICADA AO MONITORAMENTO DE ÁREAS SURPEVISIONADAS

HOLOS, vol. 1, 2011, pp. 112-136

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

Natal, Brasil

TEMPLATE MATCHING COMO TÉCNICA COMPUTACIONAL APLICADA AO MONITORAMENTO DE ÁREAS SURPEVISIONADAS

Gustavo Henrique Pereira Florentino
Mestrando Engenharia Elétrica na University of Colorado
University of Colorado – USA. E-mail: gflorent@uccs.edu
Ricardo Alexsandro de Medeiros Valentim
Professor do Departamento de Engenharia Biomédica – UFRN. E-mail: ricardo.valentim@ufrnet.br
Heitor Uchoa Bezerra
Aluno do curso de Engenharia da Computação da UFRN. E-mail: heitoru@dca.ufrn.br.
Bruno Gomes de Araújo
Doutorando do Departamento de Engenharia de Computação e Automação –UFRN. E-mail: brunogomes@dca.ufrn.br
João Marcos Teixeira Lacerda
Mestrando do Departamento de Engenharia de Computação e Automação –UFRN. E-mail: jonymac@deb.ufrn.br
Alexandre Clemente
Aluno do curso de Engenharia da Computação da UFRN.E-mail: alexandreclemente@dca.ufrn.br.

RESUMO

Esse artigo apresenta um sistema de vigilância que realiza a detecção automática de movimento, utilizando um dispositivo de captura de vídeo. A detecção automática de movimento permite que os usuários sejam notificados sobre eventos de segurança sem a necessidade de monitoramento contínuo de vídeo realizados por seres humanos. Também é a apresentada uma modelagem do sistema baseado na API Java Media Framework da Sun Microsystems, cujo objetivo é gerar streams de vídeo em tempo real do ambiente monitorado.

PALAVRAS-CHAVE: Visão Computacional, Sistema de Vigilância, Detecção de Movimento, Java Media Framework.

TEMPLATE MATCHING TECHNIQUE AS APPLIED TO MONITORING OF COMPUTATIONAL FIELDS OF SUPERVISION

ABSTRACT

This paper presents a surveillance system that performs movement automatic detection using a video capture device. Automatic detection of motion allows users to be notified of security events without the need for continuous video monitoring performed by humans. Also presented is a modeling system based on Java Media Framework API of Sun Microsystems, whose goal is to generate video streams in real-time for monitoring of environmental.

KEY-WORDS: Computer Vision, Surveillance system, Movement detection, Java Media Framework.

TEMPLATE MATCHING COMO TÉCNICA COMPUTACIONAL APLICADA AO MONITORAMENTO DE ÁREAS SURPEVISIONADAS UTILIZADO

INTRODUÇÃO

Problemas relacionados a segurança têm sido um aspecto recorrente na sociedade atual, gerando demanda por sistemas nessa área, cujo objetivo é possibilitar aos usuários o sentimento do estar seguro e deste modo permitir também a sensação de conforto e tranqüilidade. Os avanços na área da eletrônica e popularização das câmeras de vídeo, tem permitido que aplicações voltadas a segurança façam o maior uso deste tipo de dispositivo (Ziliani e Cavallaro. 1997).

Segundo Muller-Schneiders at al. (Muller-Schneiders et. al, 2005) as questões chaves para a maior adoção de sistemas com análise automática de vídeos são:

- A robustez dos sistemas disponíveis atualmente deve ser aumentada. Um alto número de falsos positivos e/ou falsos negativos não é tolerado no domínio de sistemas de segurança e;
- A falta de métricas de qualidade para algoritmos e sistemas de segurança que façam uso de visão computacional.

A primeira questão pode ser resolvida individualmente pelos desenvolvedores do sistema. Porém a segunda questão depende do esforço da comunidade de pesquisa em adotar um padrão que permita avaliar numericamente a qualidade dos algoritmos de análise de vídeo. Nessa perspectiva, o presente artigo buscou abordar estritamente a primeira questão, de modo, que os resultados apresentados demonstram uma significativa redução no número de falsos positivos e/ou falsos negativos. Portanto, demonstrando a viabilidade não apenas do sistema desenvolvido, mas também das técnicas aplicadas - fator preponderante a ser considerando pelos sistemas de segurança.

Atualmente existem vários tipos de sistema de segurança baseados em vídeo. Porém, vários deles dependem de intervenção humana para a detecção de invasões. Alguns sistemas se baseiam na gravação de vídeo localmente para uma posterior análise do evento – mais comumente chamada de análise a frio. O problema desse tipo de abordagem é que não há uma detecção do evento no momento em que ele ocorre. Em geral, nenhuma atitude pode ser tomada no exato momento da ocorrência do evento devido à falta de comunicação do sistema com um meio exterior (Ziliani e Cavallaro. 1997).

Outro tipo de sistema são os que enviam um fluxo contínuo de vídeo para uma central de segurança, porém sem uma detecção automática de variação no vídeo (mudança de cenário), se faz necessária então, a presença de um operador que visualize continuamente os vídeos e detecte a invasão no instante em que ocorrer. O grande problema desse tipo de sistema é a necessidade de atenção contínua e ininterrupta do operador. Neste contexto, o sistema é vulnerável a uma falha humana, como por exemplo: desatenção e/ou negligência. Aspectos que pode inviabilizar o uso de sistemas de vigilância quando o ambiente monitorado trata de uma grande quantidade de câmeras (Muller-Schneiders et. al, 2005).

Neste contexto, o presente artigo apresenta um sistema desenvolvido cujo objetivo foi detectar automaticamente a invasão em uma área supervisionada. Para isso, o sistema faz uso de um dispositivo de captura de vídeo (por exemplo, webcam) e realiza a amostragem da imagem do local em instantes de tempo definidos. Através de processamento das imagens obtidas, o

sistema é capaz de detectar se houve uma alteração significativa na imagem (cenário monitorado), o que caracteriza, portanto, uma possível invasão dessa área supervisionada.

Além de detectar movimentos, o sistema informa sobre os eventos ocorridos a programas clientes que podem estar fisicamente distribuídos. A arquitetura do sistema adota o modelo cliente-servidor para realizar tal comunicação. O programa servidor é responsável por realizar a captura do vídeo, a detecção de movimentos e a notificação dos eventos aos clientes que estejam conectados ao sistema. E o programa cliente é responsável por receber os eventos do servidor e disponibilizar o vídeo e as imagens ao usuário.

Com base no exposto, o presente trabalho contribui efetivamente para encontrar soluções que contornem os problemas observados anteriormente. Uma vez que na detecção automática não é necessário que um operador verifique várias horas de vídeos simultaneamente. E como todo esse processamento ocorre em tempo real, é possível se tomar uma ação imediata contra a possível ou provável invasão. Outra característica do sistema é que o investimento em equipamento para sua adoção é baixo, visto que os equipamentos utilizados são bastante acessíveis e populares.

Neste sentido, foi realizada a modelagem computacional do sistema, bem como, sua implementação para validar a arquitetura proposta, analisar os resultados obtidos e demonstrar experimentalmente a viabilidade do sistema.

PROCESSAMENTO DE IMAGENS DIGITAIS

Segundo Gonzalez (Gonzalez e Woods, 2000), uma imagem monocromática refere-se á função bidimensional de intensidade da luz f(x,y), onde x e y denotam as coordenadas espaciais e o valor de f em qualquer ponto (x,y) é proporcional ao brilho (ou níveis de cinza) da imagem naquele ponto.

Uma imagem digital é uma imagem f(x,y) discretizada tanto nas coordenadas espaciais quanto no brilho. Uma representação apropriada para imagens em sistemas computacionais são matrizes de duas dimensões. Onde as linhas e as colunas podem identificar a coordenadas y e x, respectivamente. E o valor correspondente para cada par de coordenadas representa o brilho naquele ponto (Gonzalez e Woods, 2000) e (Ahumada, 1993). Cada elemento da matriz da imagem digital denomina-se pixel.

De acordo com Jain (Jain, 1989), uma imagem pode representar a luminância de um objeto em um cenário (como as fotos tiradas por uma câmera ordinária), as características de absorção de um tecido corpóreo (a imagem de um raio-X), reflexos de ondas eletro-magnéticas (imagens de radares), medição de calor (imagens em infravermelho) ou do campo gravitacional em uma área (imagem geofísica).

Em processamento de imagens, vários passos são traçados até se atingir os resultados desejados. Nesses passos estão envolvidos hardwares, softwares e fundamentos matemáticos (Gonzalez e Woods, 2000) e (Mannos e Sakrison, 1974). Estes elementos constituem um fluxo essencial para se processar uma imagem, os quais são descritos respectivamente nas subseções seguintes (A, B, C e D).

A - AQUISIÇÃO DE IMAGENS

O primeiro passo no processo é a aquisição da imagem. Ou seja, obter a imagem necessária através de um dispositivo de entrada e digitalizá-la. O dispositivo de entrada pode ser dos mais diversos, tais como: câmeras, placas de captura de vídeo, mídias digitais, scanners, etc. Já a digitalização envolve dois conceitos distintos: a amostragem e a quantização. A amostragem refere-se à digitalização das coordenadas espaciais (x, y). Já a quantização é a digitalização da amplitude dos níveis de cinza (ou das cores primárias em um sistema RGB) (Gonzalez e Woods, 2000), (Jain, 1989). A Figura 1 (A) mostra a imagem com uma amostragem e quantização adequada para visualização humana, a Figura 1 (B) demonstra uma imagem com uma amostragem baixa, e a Figura 1 (C) exibe uma imagem com uma quantização muito baixa.

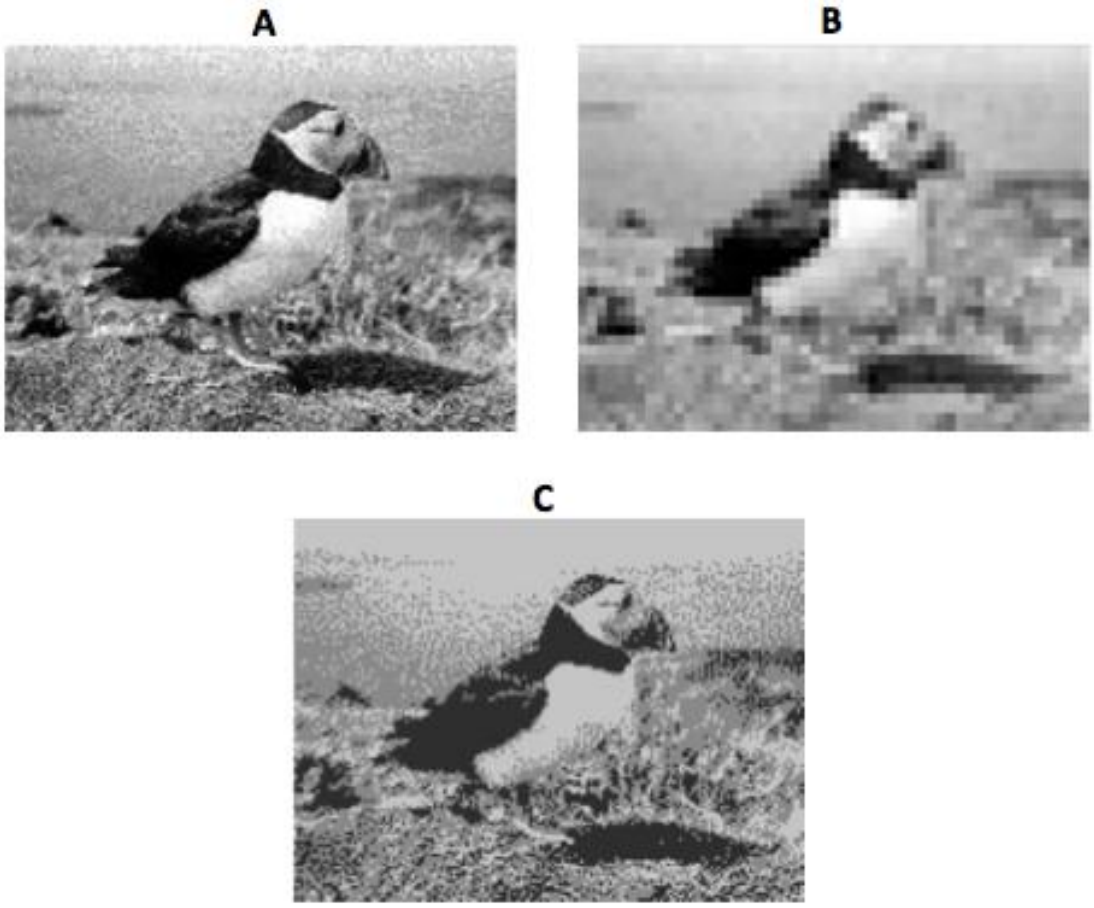

Figura 1: (A) Imagem com boa amostragem e quantização, (B) Imagem com baixa amostragem e (C) Imagem com baixa quantização.

B - PRÉ-PROCESSAMENTO

Esse passo envolve, na maioria das vezes, o melhoramento da imagem de tal forma que se tenha maiores chances de sucesso no processamento da mesma. O pré-processamento envolve técnicas de realce de contraste, remoção de ruído, conversão de esquema de cores (Gonzalez e Woods, 2000) e (Ahumada, 1993).

C - SEGMENTAÇÃO

A segmentação é o passo no qual, uma imagem é dividida em seus elementos constituintes. Por exemplo, é papel da segmentação extrair o texto e o plano de fundo de uma imagem.

Outro exemplo de segmentação seria a retirada de um padrão da imagem, como alguma figura geométrica conhecida previamente (Ahumada, 1993). A segmentação é uma das fases críticas no processamento de imagens, uma vez que um erro nesse passo se propaga para todas as fases seguintes. Por outro lado, uma segmentação bem realizada, em geral, facilita em muito os processamentos seguintes etapas (Gonzalez e Woods, 2000).

D - RECONHECIMENTO E INTERPRETAÇÃO

O processo de reconhecimento atribui rótulos aos elementos passados pelo descritor. Já a interpretação envolve a atribuição do significado a um conjunto de objetos reconhecidos. Por exemplo, os estabelecimentos das relações entre a identificação da letra Z na imagem e o caractere Z (Gonzalez e Woods, 2000).

REGISTRO DE IMAGENS

Um problema existente no processamento de imagens ocorre quando imagens obtidas a partir de diferentes posições, momentos ou sensores precisam ser comparadas (Brown, 1992). Nesse caso as imagens precisam ser alinhadas de tal forma que a comparação possa ser realizada. Para resolver esse problema existe o registro de imagens. O registro de imagens é uma transformação aplicada de tal forma que, os pixels encontrados em uma imagem sejam relacionados com os pixels de outra imagem (Brown, 1992). A transformação pode ser aplicada para corrigir deslocamentos entre imagens, diferenças de rotações, diferenças de escala e até diferenças da perspectiva de visão (Pratt, 2001). O registro de imagens é de extrema importância na comparação de imagens para que apenas as reais diferenças entre as imagens sejam detectadas (Ahumada, 1993).

De maneira mais formal, o registro de imagens pode ser definido como um mapeamento tanto espacial quanto em termos de intensidade entre duas imagens (Brown, 1992). Se definirmos as imagens como matrizes bidimensionais, I1(x,y) e I2(x,y), onde I1 e I2 mapeiam suas respectivas intensidades, então o mapeamento entre as imagens pode ser expresso como (Brown, 1992):

$$I_2(x,y) = g(I_1(f(x,y))) \qquad \text{equação (1)}$$

Onde g é uma função de transformação da intensidade da imagem I1. E f é uma função de transformação no espaço bidimensional que mapeia x e y em novas coordenadas espaciais x′ e y′, respectivamente (Brown, 1992):

$$(x', y') = f(x,y) \qquad \text{equação (2)}$$

O problema do registro de imagens é encontrar as funções de transformação g e f que melhor mapeiem o relacionamento entre os pixels de duas imagens.

MÉTRICAS DE COMPARAÇÃO DE IMAGENS

Existem várias métricas de comparação de imagens na literatura. Ahumada (Ahumada, 1993), Waston (Waston, 1993) e Peli (Peli, 1995) revisaram várias métricas utilizadas em processamento de imagens e visão computacional. Martin e Crowley (Martin e Crowley, 1995) apresentam um comparativo de métodos de correlação em sinais com diferentes tipos de ruídos. Zhou et. al (Zhou et. al, 2002) realizou um estudo comparativo de métricas mais utilizadas em comparação de imagens.

O julgamento por humanos da diferença entre duas imagens é um processo muito complexo e subjetivo. Em geral é feito em determinado contexto, focando uma ou várias características da imagem. Uma métrica de comparação computacional é altamente simplificada em relação à capacidade humana de diferenciar imagens. A medição da diferença é conceitualmente similar ao da distância. De tal forma que, uma métrica M deve seguir as regras descritas na Tabela I para quais quer imagens a, b e c (Zhou et. al, 2002).

Tabela 1: Propriedades Apresentadas pelas Métricas de Comparação.

Dicotomia	M(a,b) > 0 ou M(a,b) = 0
Identidade	M(a,b) = 0, se e somente se a = b
Comutatividade	M(a, b) = M(b, a)
Proximidade	Se M (a, b) < M (a, c), b é dito ser mais próximo de a do que c segundo a métrica M

A - COMPARAÇÃO ENTRE AS MÉTRICAS

Em seu trabalho, Zhou (Zhou et. al, 2002) utilizou uma esfera gerada computacionalmente como imagem padrão. Então foram escolhidas quatro imagens para compor o conjunto de imagens Esim: uma bola de bilhar, uma bola de *golf*, uma bola de *basket* e uma bola de tênis. Foi definido outro conjunto com quatro imagens denominado Edif . As imagens que compõe esse grupo são: um cubo gerado computacionalmente, um retrato de uma pessoa, um pote de chá e uma imagem apenas com ruído gaussiano. O conjunto formado pela união dos dois conjuntos é definido por Eall. O grupo de imagens Esim possui a característica de ter imagens que possuem similaridade de forma com a imagem padrão. Já o grupo Edif possui imagens que são descorrelacionadas com a imagem padrão.

Zhou (Zhou et. al, 2002) definiu que uma boa métrica deve possuir um desvio padrão razoavelmente alto entre os valores resultantes da comparação do padrão com cada imagem de Eall. Define-se que μdif e μsim são os valores médio retornados por uma métrica quando aplicadas as imagens do conjunto Edif e Esim, respectivamente. Então outra característica que Zhou (Zhou et. al, 2002) declarou como importante para uma boa métrica é que a diferença entre μdif e μsim deve ser alta. Define-se então que $\delta = \mu dif - \mu sim$.

A Tabela II demonstra os resultados dos desvios-padrão e as médias obtidas quando aplicadas as imagens de Esim, Edif e Eall. A Figura 2 exibe os valores sall e δ demonstrados na Tabela II. Segundo definido anteriormente, as melhores métricas possuem os mais altos valores de sall e de δ. Então, verifica-se que, as métricas como: Template Matching (TM), Comparação de Fourier de Segunda Ordem (FTT-2) e Visual Differences Predictor (VDP), possuem os melhores resultados de acordo com a pesquisa de Zhou (Zhou et. al, 2002).

Tabela 2: Média e desvios-padrão de cada métrica para cada conjunto de imagens.

Métricas	E_{sim}		E_{dif}		E_{all}		δ
	μ_{sim}	s_{sim}	μ_{dif}	s_{dif}	μ_{all}	s_{all}	
MSE	0.66	0.21	0.55	0.17	0.77	0.19	0.22
NMSE	0.66	0.25	0.46	0.11	0.85	0.26	0.39
TM	0.49	0.34	0.23	0.12	0.75	0.28	0.52
FFT-M	0.46	0.23	0.30	0.13	0.62	0.16	0.32
FFT-D	0.51	0.26	0.35	0.13	0.68	0.25	0.33
FFT-A	0.58	0.26	0.41	0.16	0.76	0.23	0.35
FFT-2	0.50	0.34	0.20	0.09	0.80	0.18	0.60
HVS-M	0.64	0.25	0.45	0.12	0.84	0.17	0.39
HVS-D	0.78	0.28	0.57	0.16	0.99	0.18	0.43
HVS-A	0.80	0.28	0.58	0.16	1.01	0.19	0.43
VDP	0.77	0.51	0.50	0.37	1.04	0.48	0.54

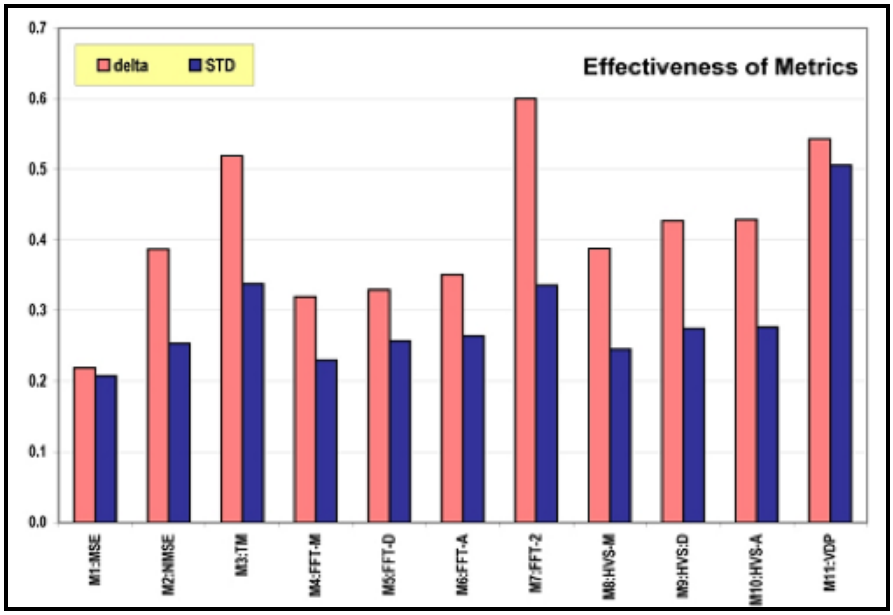

Figura 2: Gráfico com os valores de SALL (em azul) e δ (em vermelho)

B - DEFINIÇÃO DE UMA MÉTRICA COMO MÉTODO PARA DETECÇÃO DE MOVIMENTOS ON-LINE

Segundo o estudo de Zhou at. al. (Zhou et. al, 2002), foi possível verificar que três métricas apresentaram bons resultados, estas foram: Template Matching, Comparação de Fourier de Segunda Ordem e Visual Differences Predictor.
A métrica Template Matching (TM) trabalha no domínio espacial, enquanto as métricas comparação de Fourier de Segunda Ordem (FTT-2) e Visual Differences Predictor (VDP) trabalham no domínio da freqüência. Como o sistema objeto deste artigo, atua no modo on-line, necessita, portanto, de uma rápida resposta na comparação da imagens, então, a escolha natural é por métodos no domínio espacial, visto que, o processamento no domínio espacial dispensa a transformação direita e inversa para o domínio da freqüência (Mannos e Sakrison, 1974) e (Ahumada, 1996).

Com base no exposto a métrica Template Matching foi adotada como método a ser utilizado para detecção de movimentos pelo sistema objeto do presente artigo. Isso devido á sua simplicidade, eficiência e baixo custo computacional em relação ás outras métricas pesquisadas em (Zhou et. al, 2002), (Castleman, 1996) e (Daly, 1993).

JAVA MEDIA FRAMEWORK

O Java *Media Framework* é uma API (Interface de Programação de Aplicativos) multiplataforma para a exibição de mídia baseada em tempo (Sun Microsystems, 2006). A API JMF foi desenvolvida para suportar a maioria dos formatos de mídia padrão tais como: MPEG-1, MPEG-2, QuickTime, AVI, WAV, AU, and MIDI (Deitel e Deitel, 2003). A Figura 3 exibe como as camadas abstraem as peculiaridades de cada plataforma.

Os reprodutores de mídia para computadores desktop são altamente dependentes da plataforma nativa (Sun Microsystems, 2006). A API JMF provê uma abstração que esconde os detalhes de implementação dos desenvolvedores. Com isso, é possível construir aplicações multimídia sem ter o conhecimento de qual plataforma o sistema irá funcionar. Para tanto, a API JMF se baseia no conceito de DataSources.

Os DataSources são efetivamente, elementos da que encapsulam a localização, o protocolo e o software usado para transmitir a mídia (Sun Microsystems, 2006). Um DataSource pode ser instanciado a partir de um objeto do tipo MediaLocator ou através de uma URL (*Universal Resource Locator*). Uma vez adquirido, o DataSource não pode ser reutilizado para enviar outra mídia.

Os Java *Media Players* podem reproduzir os dados de mídias obtidas de várias fontes. O *framework* JMF suporta dois tipos diferentes de DataSources (Sun Microsystems, 2006) e (Deitel e Deitel, 2003):

- *Pull* Data-Source: o cliente inicia a transferência e controla o fluxo de dados do DataSource. Os protocolos mais utilizados com esse tipo de DataSource é o HTTP (*Hypertext Transfer Protocol*) e o FILE.

- *Push* Data-Source: nesse tipo de DataSource o servidor inicia a transmissão e controla o fluxo de dados. Esse tipo de DataSource compreende transmissões de mídia em multicast, unicast e vídeo sob demanda. Para transmissões em broadcast, geralmente é utilizado o protocolo RTP.

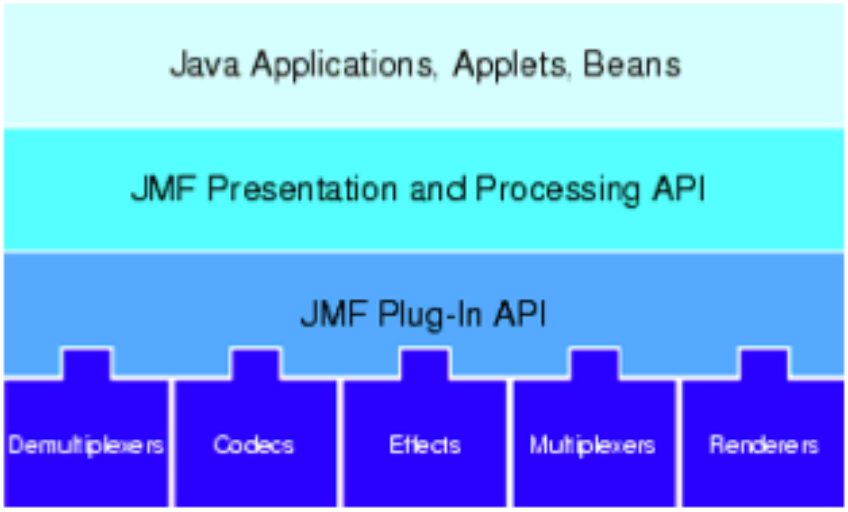

Figura 3: Arquitetura em Alto Nível do Framework JMF

A. *PLAYER*

Um Java *Media Player* é um objeto que recebe dados de um DataSource, e os renderizam de forma sincronizada (Sun Microsystems, 2006). O Java Media Player implementa a interface *Player*. Devido o processamento de mídia geralmente consumir muitos recursos, vários dos métodos utilizados na JMF são não bloqueantes e permitem uma notificação assíncrona das mudanças de estado através eventos (Olson, 2002). Por exemplo, antes de um *Player* ser iniciado, ele deve antes ter passado pelos estados *Prefetched* e *Realized*. O *framework* JMF define seis estados para os *Players*: *Unrealized, Realizing, Realized, Prefetching, Prefetched* e *Started*, a Figura 4 ilustra os estados que um *player* pode assumir. Devido à mudanças de estado às vezes levarem tempo, uma aplicação JMF pode carregar uma *thread* para instanciar um *Player* enquanto outras operações são realizadas em paralelo. Quando o *Player* estiver no estado *Prefetched*, então pode-se iniciar a reprodução do conteúdo (Deitel e Deitel, 2003).

Na Figura 4 são exibidos os 6 estados que um player JMF pode assumir, os quais são descritos a seguir:

1. *Unrealized*: esse é o estado no qual o *Player* entra assim que é instanciado.

2. Realizing: ao se chamar o método *realize()* do objeto *Player*, o estado passa a ser *Realizing*. Nesse estado, o *Player* está determinando os seus recursos necessários para reproduzir a mídia.

3. *Realized*: ao se atingir esse estado, o *Player* sabe quais os recursos necessários e o tipo de mídia que será apresentada.

4. *Prefetching*: o objeto *Player* entra nesse estado ao se invocar o método *prefetch()*. Nesse estado, o Player se prepara para reproduzir a mídia e carrega a mídia que será apresentada.

5. *Prefetched*: esse estado informa que o carregamento da mídia foi concluído e está pronto para reproduzir a mídia.

6. *Started*: nesse estado a mídia passa a ser reproduzida. O *Player* entra nesse estado ao se invocar o método *start()*.

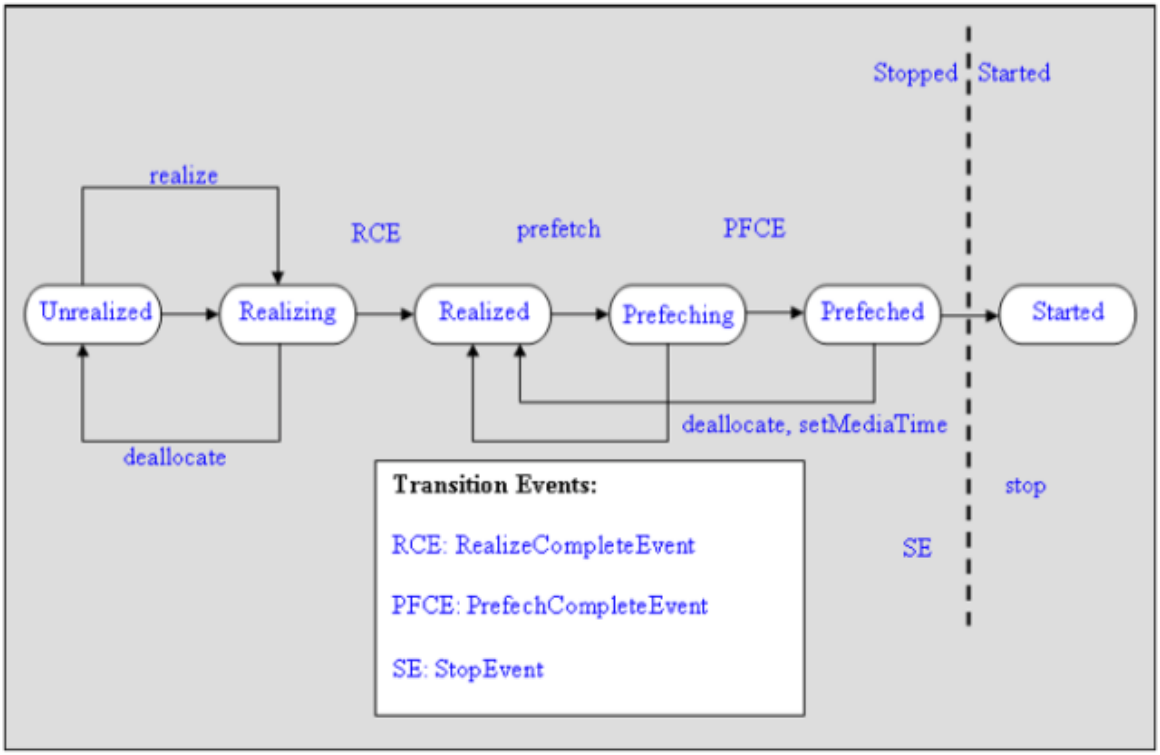

Figura 4: Ilustra os estados que um *Player* pode assumir

B. PROCESSOR

O *Processor* é um tipo de *Player*. Na API JMF, a interface *Processor* extende *Player*. Diferentemente de um *Player*, o *Processor* tem o controle sobre o que está sendo processado na mídia de entrada (Java World, 2006). Além de renderizar os dados de entrada, o *Processor* pode retornar dados de mídia através de um DataSource, desta forma ele pode ser representado por outro *Player* ou *Processor*, processado por outro *Processor* ou então convertido para outro formato. Os *Processors* possuem mais dois estados: *Configuring* e *Configured*.

- *Configuring*: o Processor entra nesse estado quando se chama o método *configure()*. O Processor só passa para outro estado quando ele termina de se conectar ao DataSource, demultiplexar a mídia e receber informações sobre o formato da mídia.

- *Configured*: o *Processor* chega a esse estado depois que as operações de *Configuring* forem finalizadas.

C. MANAGER

O *Manager* é um objeto intermediário que integra as implementações de interfaces que são usadas de forma uniforme com classes existentes. Existem quatro classes *Manager*, cada uma com uma função distinta:

- *Manager*: é a classe utilizada para criar *Players*, *Processors*, DataSources e DataSinks.

- PackageManager: essa classe mantém o registro dos pacotes que contém classes JMF.

- CaptureDeviceManager: classe que possui o registro de todos os dispositivos de captura disponíveis no sistema.

- PlugInManager: contém o registro dos plug-ins de processamento disponíveis no *framework*, tais como: *Multiplexers*, *Demultiplexers*, *Codecs*, *Effects* e *Renderers*.

REAL-TIME TRANSFER PROTOCOL

O *Real-Time Transfer Protocol* (RTP) é um protocolo desenvolvido pela *Internet Engineering Task Force* (IETF) que provê serviços de transmissão fim-a-fim de dados com características de tempo real (Schulzrinne et. al, 1996). A maioria das aplicações distribuídas que realizam a transmissão de dados em tempo real, tais como vídeo e áudio, fazem uso do protocolo RTP (Pajares et. al, 2002).

O protocolo RTP não previne a entrega fora de ordem, nem provê a sincronia na entrega do conteúdo, ou outras garantias de qualidade de serviço. Para isso, o RTP encarrega protocolos de outras camadas para resolver essas questões. Os números de seqüência recebidos pelo receptor são usados para remontar os dados na ordem correta (Pajares et. al, 2002) e (Schulzrinne et. al, 1996).

Em geral, as aplicações em RTP trabalham em cima do protocolo UDP com o fim de aproveitar os serviços de multiplexação e de checagem de checksum (Schulzrinne et. al, 1996). Protocolos orientados a conexão, como o HTTP que é baseado no TCP, geralmente não são utilizados em transmissão de mídia devido ao overhead que as conexões confiáveis requerem pela retransmissão de pacotes perdidos. O atraso devido a retransmissão pode degradar a transmissão de um vídeo ao vivo, e essa é uma conseqüência inaceitável em aplicações em tempo real. O UDP não garante a chegada dos pacotes ao seu destino, porém essa é uma limitação admissível em um vídeo que tolera a perda de alguns pacotes (Wilcox, 2000).

O pacote RTP consiste de um cabeçalho RTP fixo, uma lista das fontes contribuintes (fontes de streams RTP que compõe o pacote) e os dados que se deseja enviar (Schulzrinne et. al, 1996). Sessão RTP é a associações entre um conjunto de participantes se comunicando com RTP (Schulzrinne et. al, 1996). Uma sessão RTP é definida para cada participante como um par composto de endereço de destino e porta. O endereço de destino pode ser igual para todos os participantes no caso de um IP multicast, ou pode ser diferente para cada um no caso de uma rede unicast com a porta sendo comum para cada um.

Juntamente com o RTP, existe o protocolo de controle RTCP (RTP *control protocol*, RTCP) que tem como objetivo prover uma resposta sobre a qualidade dos dados distribuídos. O RTCP é baseado na transmissão periódica de pacotes de controle para todos os participantes da sessão. O protocolo sobre o qual o RTP trabalha deve prover multiplexação de pacotes de dados e controle. No caso do UDP, isso é possível fazendo-se uso de portas diferentes para os pacotes de controle e de dados. Esse protocolo faz parte do papel do RTP como protocolo de transporte e trata do fluxo de dados e do controle de congestionamento. Esse protocolo pode ser útil para algoritmos de codificação adaptativos que aumentam a taxa de compressão de acordo com o aumento do congestionamento da rede (Schulzrinne et. al, 1996).

A Figura 5 demonstra a arquitetura do utilizada pelo protocolo RTP e o seu relacionamento com os outros protocolos.

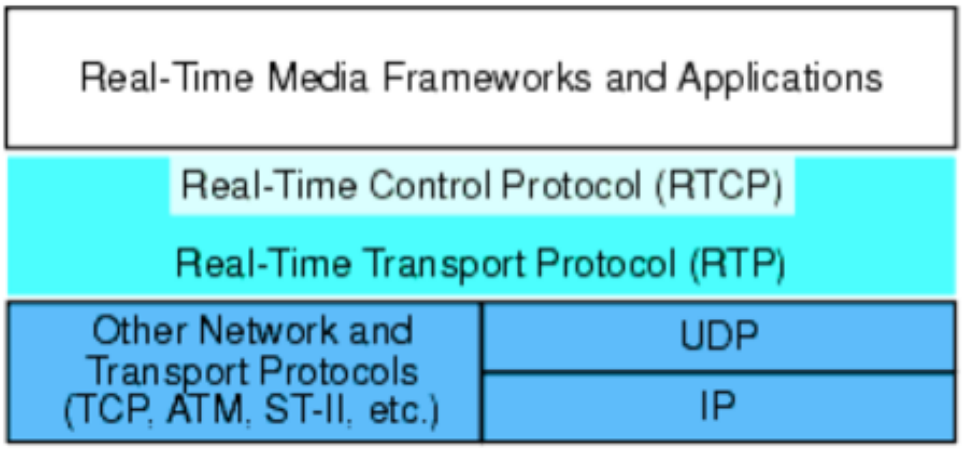

Figura 5: Arquitetura do Protocolo RTP

SEGURANÇA COM DETECÇÃO AUTOMÁTICA DE MOVIMENTOS UTILIZANDO VISÃO COMPUTACIONAL

Esta seção do artigo trata da modelagem e implementação do sistema de segurança proposto, também foi feita a escolha da métrica de comparação que melhor atende aos requisitos do sistema.

Para o desenvolvimento do sistema, foram utilizadas as seguintes ferramentas:

- Java Development Kit 6.0;
- Java Media Framework 2.1.1e;
- Eclipse SDK 3.1;
- JDesktop Integration Components (JDIC) versão 0.9, Release 1;
- Duas webcams.

A. MÓDULOS DO SISTEMA

O sistema proposto realiza a detecção automática de movimentos através de imagens capturadas por uma câmera. Além disso, o sistema deve permitir que o usuário se conecte remotamente ao sistema para poder ser notificado dos eventos ocorridos.

Verifica-se que o sistema computacional envolve dois módulos distintos: o módulo servidor e o módulo cliente. O módulo servidor é responsável por obter as imagens do dispositivo de captura de vídeo, realizar o processamento de imagem para detectar os eventos, disponibilizar o vídeo para os clientes e informar os clientes de eventos ocorridos. Já o módulo cliente tem a função de se conectar ao servidor com o objetivo de ser notificado dos eventos ocorridos no local supervisionado e de receber o vídeo disponibilizado pelo servidor.

A Figura 6 apresenta a arquitetura física do sistema, demonstrando que quando ocorre um movimento detectado pelo servidor, o cliente é informado sobre esse evento. Devido à distinção evidente entre os dois módulos, num primeiro momento será abordado o módulo servidor, depois o módulo cliente e por fim será tratada a associação entre os dois módulos.

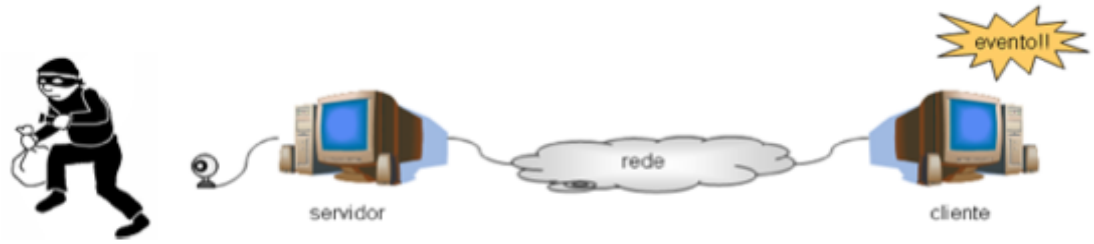

Figura 6: Arquitetura Física do Sistema de Segurança

B. MÓDULO SERVIDOR

Os objetivos principais do módulo servidor estão listados a seguir:

- Obter as imagens do dispositivo de captura e detectar os eventos ocorridos;
- Permitir que clientes remotos se conectem ao sistema;
- Disponibilizar o vídeo do local aos clientes remotos;
- Informar aos clientes autenticados sobre a ocorrência de eventos;
- Prover ao usuário uma interface simples que permita ajustar as configurações mais importantes do sistema de acordo com a preferência do usuário.

De acordo com cada objetivo, é possível se analisar a arquitetura do módulo servidor. Cada objetivo destacado tem uma função distinta em relação ao outro, portanto pode-se modelar o módulo servidor como sendo composto dos seguintes sub-módulos:

- Servidor de Autenticação: servidor responsável pela autenticação os clientes externos.
- Servidor RTP: servidor que tem por objetivo disponibilizar o vídeo aos clientes;
- Serviço de Detecção: representa o serviços principal do sistema. Realizando o processamento de imagens e detectando os movimentos ocorridos;
- Produtor de Eventos: componente de comunicação que notifica os clientes dos eventos ocorridos e;
- Interface Gráfica: componente responsável pela interação entre o sistema e o usuário.

A Figura 7 apresenta a arquitetura proposta para o módulo servidor, além dos sub-módulos já comentados anteriormente, existe o componente Servidor que é responsável pelo controle e delegação de atividades aos sub-módulos.

Figura 7: Arquitetura Física do Sistema de Segurança

As seções seguintes abordam a modelagem computacional de cada sub-módulo separadamente. Posteriormente será discutida a associação de cada sub-módulo no módulo servidor.

O servidor de autenticação tem por objetivo verificar se o usuário que está tentando acessar o sistema possui autorização para o mesmo. O servidor de autenticação foi implementado através de um *socket* servidor. O mesmo foi criado através da classe *java.net.ServerSocket* disponível na plataforma J2SE.

O objeto instanciado da classe *ServerSocket* passa a ouvir uma porta esperando a conexão de um cliente. Quando recebe uma conexão é criado um *socket* e através desse *socket* o servidor recebe uma string contendo:

nome do usuário: senha: porta de eventos

O nome do usuário e senha são checados no banco de dados. O número da porta de eventos informa ao servidor qual porta será utilizada para o envio dos eventos. O servidor de autenticação retorna uma mensagem ao cliente informando o sucesso ou não do processo de autenticação. Além de informar ao cliente, o servidor de autenticação cria um objeto Sessão que é passado ao Servidor de tal forma que o Servidor tome conhecimento do novo usuário conectado ao sistema. O diagrama de classes para o sub-módulo servidor de autenticação pode ser visto na Figura 8.

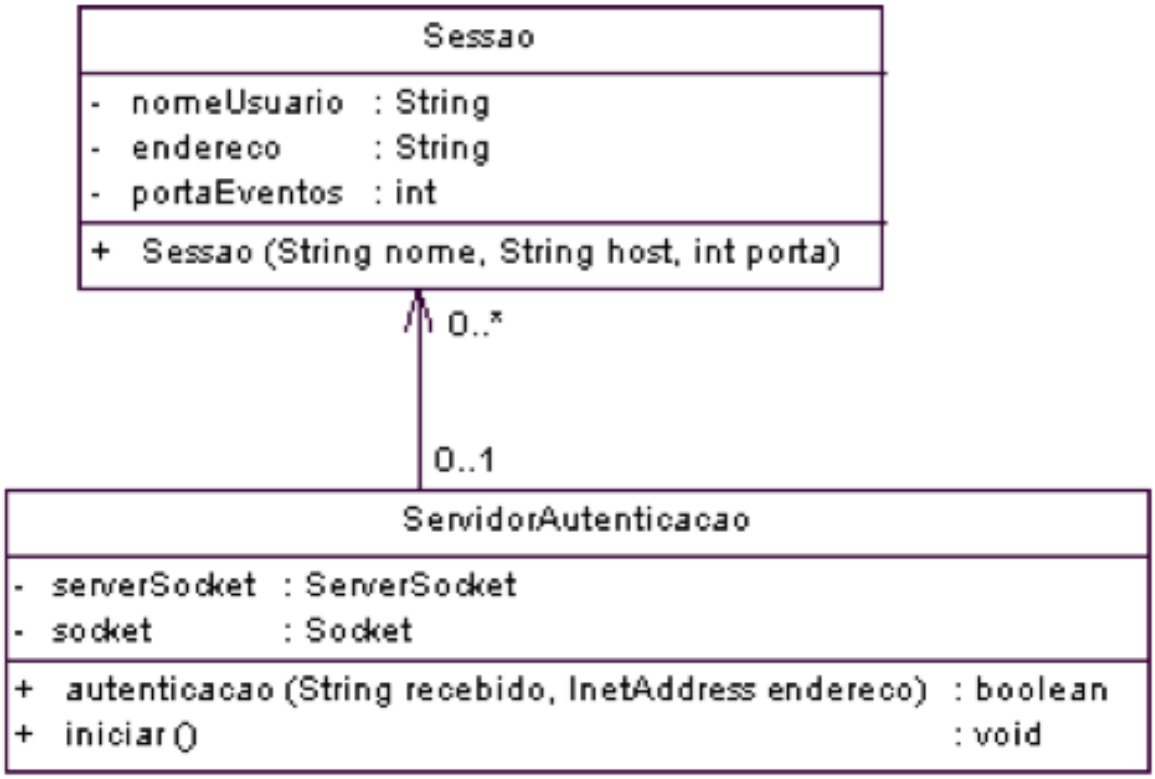

Figura 8: Diagrama de Classes do Servidor de Autenticação

C. SERVIDOR RTP

A classe ServidorRTP é uma classe criada a partir da classe VideoTransmit que é disponibilizada pela Sun Microsystems. A função dessa classe é disponibilizar o vídeo do dispositivo de captura para o cliente que esteja conectado ao sistema. Essa classe recebe o objeto DataSource do dispositivo de captura e cria um DataSink que disponibiliza o vídeo para o endereço IP do cliente conectado.

A compressão utilizada pelo Servidor RTP é a JPEG devido a ela ser ampla- mente suportada em todas as plataformas e por possuir a menor carga de dados a serem transmitidos em RTP (McCown, 2002). A qualidade da compressão do vídeo pode ser definida pelo cliente. De acordo com a classe *javax.media.control*. QualityControl, a qualidade do vídeo deve variar em uma escala de 0.0 a 1.0 (Sun Microsystems, 2006).

D. SERVIÇO DE DETECÇÃO

O serviço de detecção constitui o sub-módulo que realiza o processamento de imagens para a detecção de movimentos. Esse serviço de detecção de movimentos provê a funcionalidade básica do sistema. Como primeiro desafio, foi necessária a definição da métrica de comparação de imagens, que foi resolvida através de uma pesquisa, a qual foi fundamental na definição da métrica a ser aplicada. Especificamente para o caso da aplicação desenvolvida e apresentada neste artigo, conforme já mencionado, a métrica escolhida foi *Template Matching* visto que atende aos requisitos de uma aplicação de tempo real.

E. MÉTODO DE EXECUÇÃO

Segundo Ziliani (Ziliani e Cavallaro. 1997), em várias aplicações de segurança, o método de detecção de movimentos é baseado na comparação das imagens obtidas com uma imagem de

referência. Esse método é aplicável, pois, geralmente é possível obter uma imagem de referência do local, sem que haja objetos se movimentando em primeiro plano. Com essa abordagem é possível detectar novos objetos na cena, mesmo que eles estejam parados. Também é possível detectar novos objetos que tenham sido removidos do cenário de referência.

Ainda segundo Ziliani (Ziliani e Cavallaro. 1997), essas são características importantes na maioria das aplicada a segurança. Portanto, o método utilizado no sistema é inicialmente obter uma imagem padrão da cena e depois comparar a imagem atual da mesma com o padrão. A Figura 9 exibe o fluxograma para detecção de movimentos.

A comparação com o padrão é o ponto em que a métrica de comparação passa atuar. Para a métrica são passadas as duas imagens, a imagem padrão e a imagem atual, então o *Template Matching* calcula o grau de similaridade entre as duas imagens. Caso seja detectado uma variação acima de um limiar definido é registrada a ocorrência de um evento de segurança.

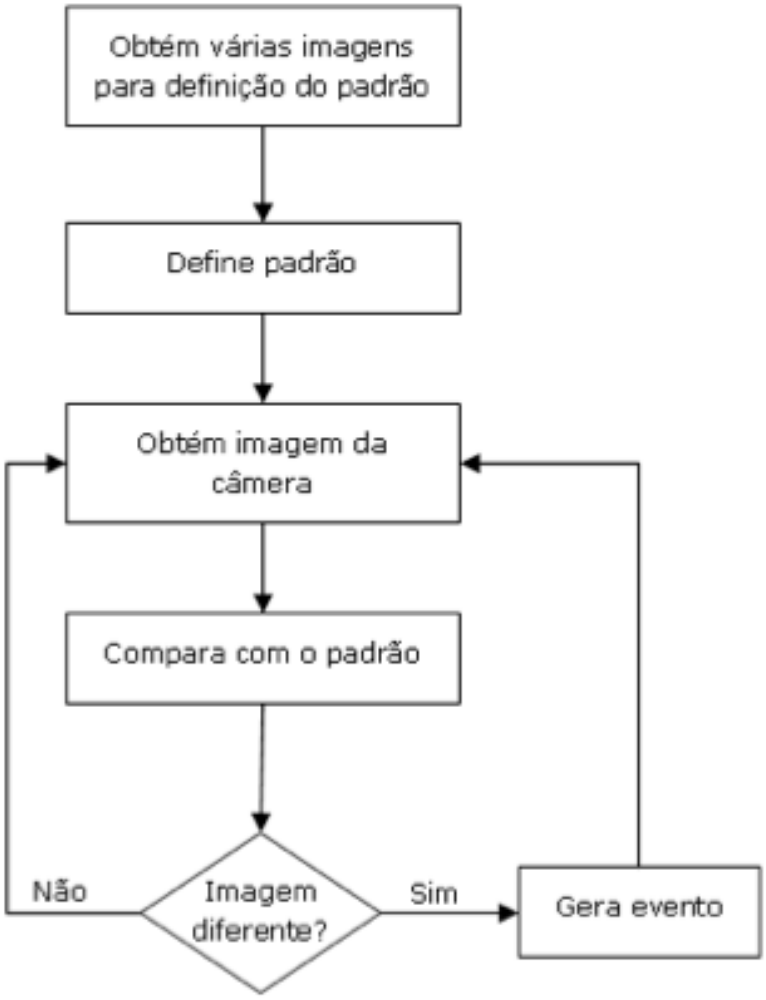

Figura 9. Fluxograma para Detecção de Movimentos

F. MÉTODO DE EXECUÇÃO

Segundo Muller-Schneiders at al. (Muller-Schneiders et. al, 2005), um sistema de segurança robusto deve gerar um baixo número de falsos positivos uma vez que o usuário do sistema pode se distrair ao receber vários alarmes sobre eventos irrelevantes como pequenas movimentações na câmera, mudanças na iluminação, ruídos na câmera, etc. Porém, simplesmente tornar o sistema mais tolerável a pequenas variações pode resultar em um problema ainda maior: a ocorrência de

falsos negativos. Como um evento de segurança que passe desapercebido pode causar sérias ameaças de segurança ao local supervisionado, então, idealmente, o sistema não deve deixar de alertar ao usuário sobre um evento relevante. Uma das maneiras de se evitar a ocorrência de falsos-positivos e falsos-negativos é uma definições precisa do padrão e do limiar de variação.

O primeiro passo para a definição do padrão é a captura de imagens da cena. Nesse momento, a imagem recebida pela câmera deve estar estática, pois ela será utilizada como padrão. Caso haja muita variação, é disparada uma exceção e o sistema reinicia o processo de definição do padrão novamente. Número de imagens obtidas para a definição do padrão é definido pelo usuário. Depois de obtidas as imagens é calculada a média μ e desvio-padrão σ do conjunto de imagens. A média das imagens é utilizada como imagem padrão para a comparação das imagens. E o desvio-padrão é utilizado para se verificar se houve movimentação durante a definição do padrão e também para servir como estimativa da qualidade da imagem gerada pelo dispositivo de captura. Caso o desvio-padrão seja alto, então o sistema passa a ser mais tolerante com relação às variações na imagem com o intuito de evitar falsos-positivos.
O método *template matching* recebe uma imagem como parâmetro e realiza a comparação da imagem com o padrão e retorna um *array* do tipo *double* com o cálculo para cada cor. O *template matching* retorna um valor entre zero e um. Onde o valor um indica perfeita igualdade entre as imagens, enquanto o valor zero corresponde a uma completa descorrelação entre as imagens.

A definição do limiar poderia ser feita diretamente através do grau de correlação retornado pelo método templateMatching, porém câmeras diferentes possuem qualidades de imagens diferentes. A qualidade da imagem pode ser referida como o nível de ruído embutido na imagem capturada, a resolução, o formato da imagem, a qualidade na calibração automática da imagem, etc. Portanto, o limiar torna-se diferente para cada câmera utilizada. Com o objetivo de tornar a definição do limiar independe da câmera, é feita a normalização da distribuição dos valores retornados pelo *template matching*. Com a normalização, faz-se que a distribuição tenha média nula e desvio padrão unitário. A normalização de distribuições normais é feita a partir da seguinte fórmula (Triola, 1998):

$$tmn = \frac{tm - \mu}{\sigma}$$ equação (3)

Onde μ é a média e σ o desvio padrão da medição retornada pela métrica *template matching* calculados durante o processo de definição do padrão, também o valor retornado pelo método templateMatching e também o seu valor normalizado (Soong, 2004) e (Spiegel, 1993). Portanto, com a normalização, a defini do limiar passa a ter um valor absoluto, independente da qualidade da imagem obtida pelo dispositivo de captura de imagens.

G. INTERFACE GRÁFICA

A interface gráfica do sistema foi implementada em *Swing* procurando mantê-la o mais simples possível. A Figura 10 demonstra a interface criada.

A tela exibe o vídeo obtido do dispositivo de captura e à direita estão as opções disponíveis ao usuário, tais como definição da sensibilidade na detecção da variação no vídeo. Permitindo que o usuário defina se deseja ser informado de qualquer mínima variação ou apenas de variações altas. Outra opção disponível ao usuário é o agendamento do início e término da execução do serviços de detecção de movimentos. O botão "Iniciar Servidor de Conexões Externas" inicia o servidor de autenticação para permitir que clientes remotos se conectem ao sistema. O botão "Iniciar

Detecção de Movimentos" inicia o serviços de detecção de movimentos permitindo que o sistema gere eventos ao ocorrer variações no vídeo capturado.

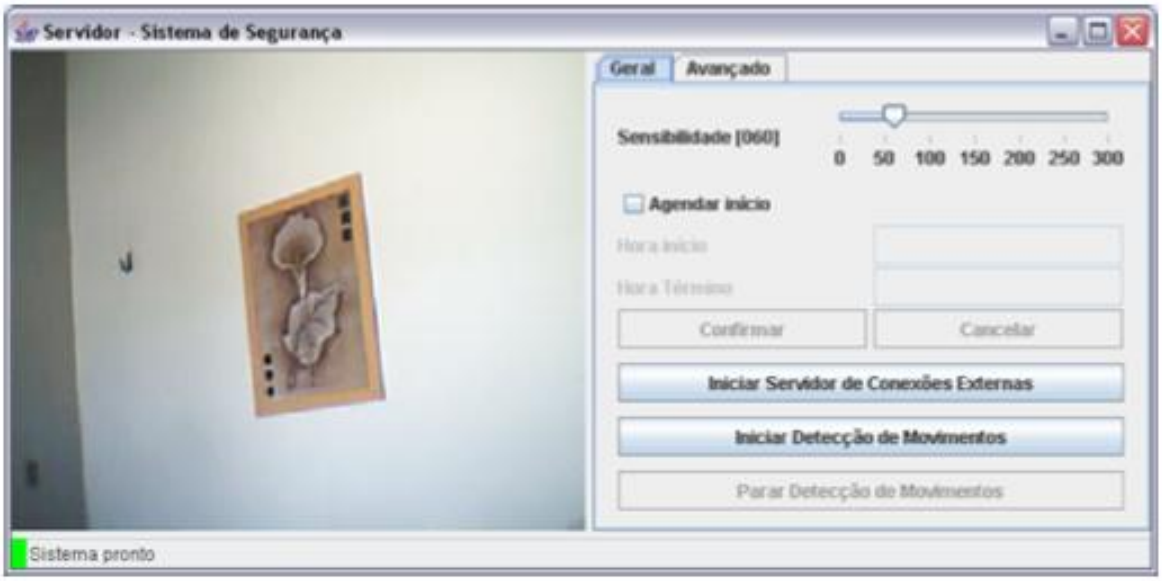

Figura 10. Interface Gráfica do Módulo Servidor

H. MÓDULO CLIENTE

O módulo cliente permite que o usuário se conecte remotamente ao servidor, sendo capaz de visualizar o vídeo do local supervisionado e de ser notificado dos eventos ocorridos. De forma similar ao servidor, o cliente foi modelado em sub-modulos.

A Figura 11 demonstra os sub-módulos que compõe o módulo cliente. Na Figura 11 existe o cliente de autenticação que se comunica diretamente com o servidor de autenticação. O sub-módulo consumidor de eventos que recebe os eventos enviados pelo produtor de eventos do módulo servidor. E a interface gráfica do usuário para a exibição do vídeo, imagens e mensagens de evento.

Figura 11. Arquitetura do Módulo Cliente

I. CLIENTE DE AUTENTICAÇÃO

A função do cliente de autenticação é enviar os dados de nome do usuário, senha e porta de eventos para o servidor com a finalidade de autenticar o usuário, para tanto, é enviada uma string através de *socket* conforme definido na seção 7.2 (Módulo do Servidor).

Ao ser autenticado, o cliente recebe uma mensagem de confirmação do servidor e então é aberta a janela principal do programa. Nesse momento é criado um objeto da classe *javax.media.Player* para receber o vídeo do servidor RTP, também de forma simultânea é criado o consumidor de eventos para receber os eventos enviados pelo servidor.

J. CONSUMIDOR DE EVENTOS

O consumidor de eventos é o sub-módulo que ouve uma porta de comunicação a espera da chegada de eventos enviados pelo produtor de eventos. O consumidor de eventos abre um *java.net.ServerSocket* para receber os objetos da classe Evento enviados pelo servidor. Ao chegar dados pelo *socket*, o objeto Evento é repassado para a classe de controle Cliente que irá encarregar a interface gráfica da exibição do evento na tela do usuário.

K. INTERFACE GRÁFICA

A interface gráfica do módulo cliente é composto por três componentes:

- Player JMF para exibição do vídeo em tempo real;
- Frame com a imagem do evento ocorrido;
- Conjunto de eventos ocorridos.

Na Figura 12 verifica-se no canto superior esquerdo a presença do player JMF que está recebendo o vídeo em tempo real do servidor RTP. Abaixo do player, está é uma janela para exibição do evento ocorrido. E à direita está um painel com os eventos ocorridos. Ao clicar em cada bloco de evento, a imagem correspondente ao evento é exibida na janela de exibição de eventos.

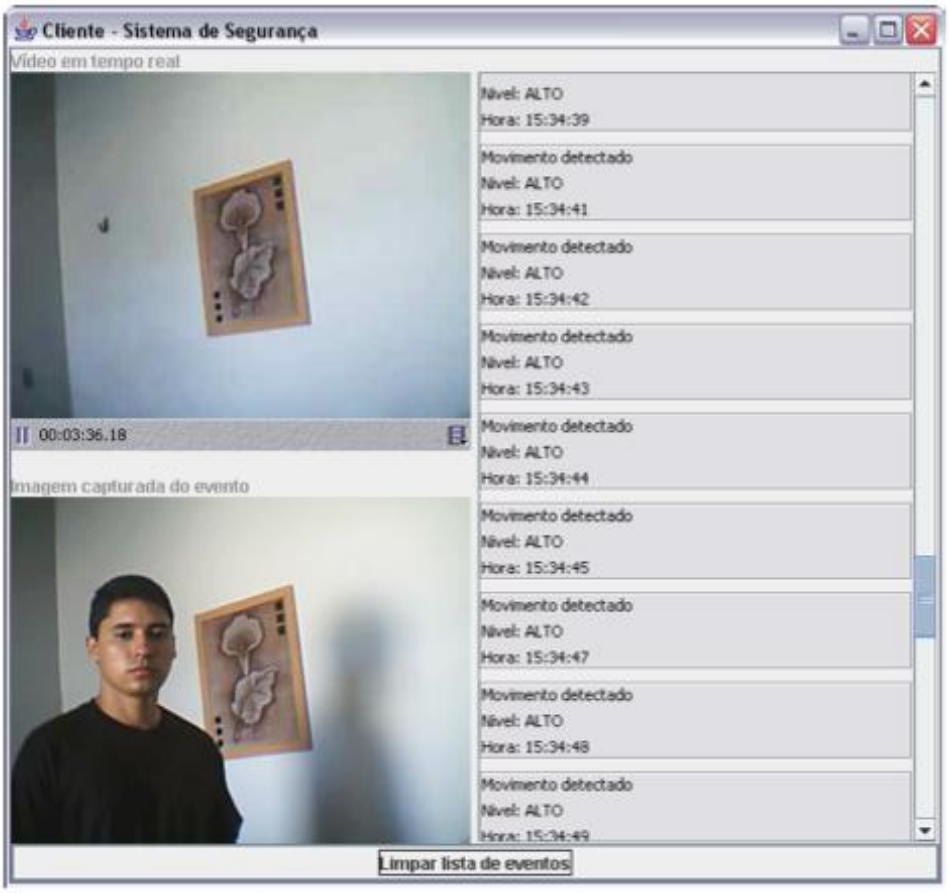

Figura 12. Interface Gráfica do Módulo Cliente

Vale também destacar que o usuário pode minimizar a tela do cliente e continuar o seu trabalho normalmente, e assim que ocorrer um evento é exibido um balão na barra de *tray* do sistema operacional. A Figura 13 ilustra a ocorrência de um evento. Essa funcionalidade da interface gráfica foi implementada fazendo-se uso da biblioteca JDesktop *Integration Components* (JDIC) versão 0.9 (Sun Microsystems, 2006) desenvolvida pela Sun Microsystems. Essa biblioteca é independente de plataforma, não comprometendo a portabilidade do sistema.

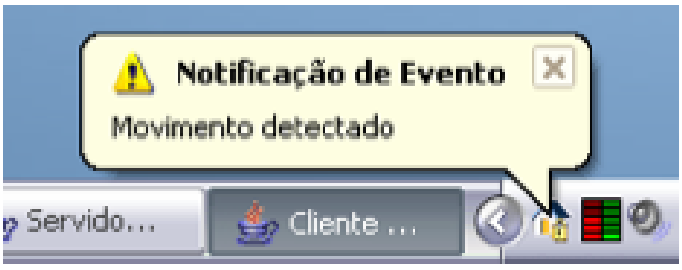

Figura 13. Ao ocorrer um evento, é exibido um alerta no *tray* do sistema operacional

RESULTADOS

Nessa seção são apresentados os resultados de testes realizados sobre o sistema implementado. Os testes realizados são baseados em duas situações: com iluminação artificial e iluminação natural. O valor utilizado para o limiar de tolerância nos ensaios é de 60. Esse é o valor default ao se iniciar o sistema. Além disso, nos experimentos foi utilizada uma taxa de dois quadros por segundo para análise dos movimentos.

A. OCORRÊNCIA DE FALSOS POSITIVOS

Verificou-se que em ambientes com iluminação constante, o sistema demonstrou-se bastante robusto. O experimento realizado consiste em manter um cenário sem nenhuma variação na iluminação artificial ou de qualquer objeto que possa provocar uma variação na imagem. São analisadas 3000 amostras num período de 25 minutos para se verificar a ocorrência ou não de falsos positivos. Este experimento foi realizado três vezes, portanto com 9000 amostras e em nenhum momento foi detectado um falso positivo.

B. VARIAÇÃO NA CAPTURA DA IMAGEM

Um efeito observado no sistema é que mesmo sem haver variação do cenário, a tendência é de ocorrer um contínuo crescimento da diferença entre as imagens capturadas e o padrão. Foram feitos ensaios nos mesmos moldes realizado na seção anterior, com 3000 amostras em 25 minutos e com o ensaio repetido três vezes. Porém dessa vez são analisados os valores (Z) retornados pela métrica de comparação. Como há flutuação no valor de Z de uma amostra para outra, é feita a média entre os 25 valores anteriores de Z e plotado em um gráfico. Portanto, cada unidade no eixo das abscissas corresponde a 25 amostras. O gráfico da Figura 14 representa a média obtida na execução do experimento.

Uma possível explicação para esse fato se fundamenta no funcionamento do dispositivo de captura utilizado. As *webcams* possuem um sistema de calibração de exposição à luz automático. Esse sistema é bastante sensível à mínimas variações da luz ambiente. Portanto com o passar do tempo, a câmera automaticamente altera a sua exposições à luz, alterando a imagem em relação ao padrão.

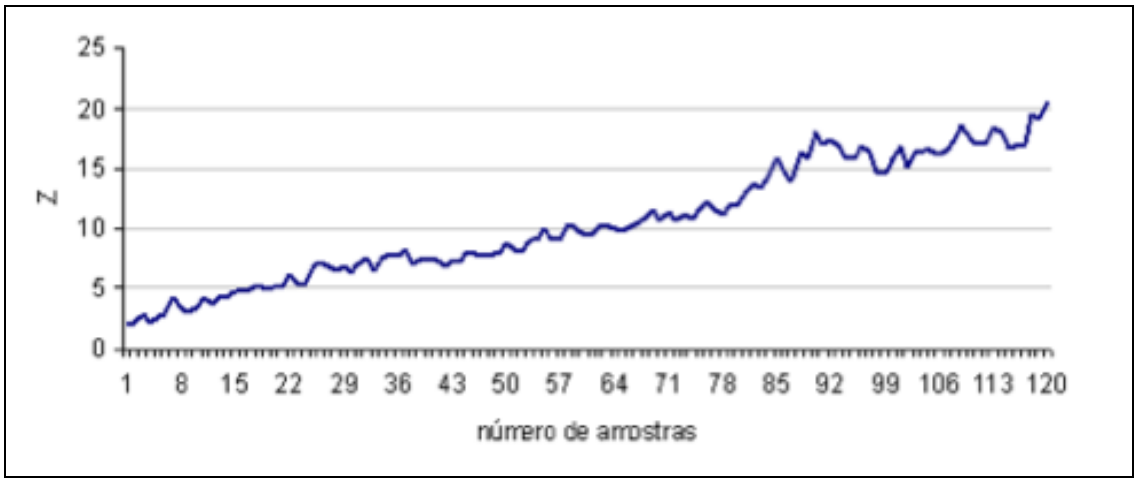

Figura 14. Evolução dos valores retornados pela métrica em função do tempo em ambiente com luz artificial

Devido a esse efeito, constatou-se que as imagens capturadas tendem a divergir quase linearmente do padrão com o passar do tempo. Para evitar que a divergência cresça a ponto de gerar falsos positivos, foi implementada a seguinte técnica para o tratamento desse problema. A cada 25 amostras comparadas, o sistema realiza a média dos valores retornados pela métrica de comparação. Caso a média do bloco de 25 amostras esteja acima de 75% do limiar de alarme de movimento, então o sistema realiza a redefinição do padrão automaticamente. Com isso, a métrica volta a detectar baixos valores de variação entre as amostras e o padrão, evitando que a divergência entre as imagens e o padrão aumente indefinidamente.

Uma possível explicação para esse fato se fundamenta no funcionamento do dispositivo de captura utilizado. As *webcams* possuem um sistema de calibração de exposição à luz automático.

Nesse sistema é bastante sensível às mínimas variações da luz ambiente. Portanto com o passar do tempo, a câmera automaticamente altera a sua exposição à luz, alterando a imagem em relação ao padrão.

No gráfico da Figura 15 verifica-se que em um ambiente sem iluminação controlada o comportamento dos valores de Z em função do tempo não demonstra a tendência linear observada na Figura 14. Isso ocorre devido à iluminação natural ser variável com o passar do tempo. Por isso verifica-se a ocorrência de dois picos que ultrapassam o limite de 75% do limiar de detecção, ou seja, 45. Vale ressaltar que logo após a ocorrência desses picos, os valores voltam a ser baixos uma vez que a redefinição do padrão foi aplicada.

Figura 15. Evolução dos valores retornados pela métrica em função do tempo em ambiente com luz artificial

C. TAXA DE ACERTO EM RELAÇÃO À DISTÂNCIA

Outro experimento realizado diz respeito à relação entre a precisão na detecção de movimentos e a distância do elemento gerador do movimento para a câmera. Neste ensaio, para cada distância são realizadas 20 movimentação que o sistema deve capturar. A movimentação consiste no caminhar de uma pessoa de estatura mediana na distância especificada. Este experimento objetiva analisar a ocorrência de falsos positivos, falsos negativos e detecção correta de acordo com a distância. O experimento foi realizado até uma distância máxima de 10 metros.

De acordo com o observado no gráfico da Figura 16, o sistema consegue detectar os movimentos corretamente até à distância máxima do teste, 10 metros. Porém verifica-se uma pequena taxa de erro na distância de um metro. Isso ocorre devido à grande proximidade da pessoa perante a câmera ocultando significativamente a iluminação incidente na câmera. Com isso, a câmera altera a sua exposição à luz. Neste sentido, depois da mudança de cena, a câmera leva alguns segundos até voltar à exposição à luz anterior. Em decorrência dessa recalibragem, as imagens capturadas após o movimento passam a gerar falsos positivos por um breve momento.

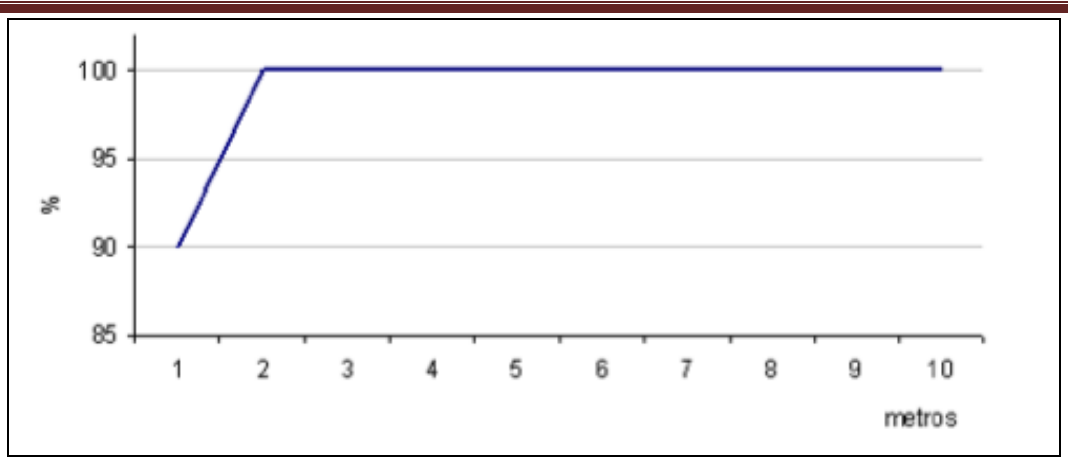

Figura 16. Relação entre distância e porcentagem de acerto na detecção de movimentos

CONSIDERAÇÕES

Com esse trabalho verificou-se que é possível utilizar um sistema de segurança com baixo investimento em equipamentos e que possui vantagens de notificação on-line dos eventos ocorridos. Permitindo que o usuário mesmo não estando presente no local supervisionado tome conhecimento de qualquer evento de segurança relevante. Uma das maiores dificuldades encontradas foi devido à calibração automática da exposição à luz das webcams encontradas no mercado. Essa calibração automática pode gerar imagens com iluminações diferentes, podendo gerar falsos positivos.

O sistema implementado possui as funcionalidades básicas para o propósito para o qual foi concebido. Porém, o sistema permite uma grande gama de trabalhos futuros a respeito dele. Tais como implementar um cliente para dispositivos móveis, implementar o sistema para receber o processamento de várias câmeras, implementar um cliente que tenha maior controle sobre as operações do servidor (por exemplo, permitir que o cliente remoto redefina o padrão, altere o limiar de sensibilidade, etc*).*

AGRADECIMENTOS

Ao CNPq, CAPES, Laboratório de Sistemas Inteligentes (LABSIS), Laboratório de Automação Hospitalar e Bioengenharia (LAHB) da Universidade Federal do Rio Grande do Norte (UFRN) e do LAIS (Laboratório de Inovação Tecnológica em Saúde) do Hospital Universitário Onofre Lopes (HUOL) da Universidade Federal do Rio Grande do Norte.

REFERÊNCIAS BIBLIOGRÁFICAS

1. AHUMADA, A. J. **Simplified vision models for image quality assessement.** SID International Symposium Digest of Technical Papers, 1996.
2. AHUMADA, A. J. **Computational image quality metrics: a review.** SID International Symposium Digest of Technical Papers, 1993.
3. BROWN, L. G. **A survey of image registration techniques.** ACMComputing Surveys, Dezembro 1992.
4. CASTLEMAN, K. R. **Digital Image Processing**. Prentice Hall, 1996.
5. DALY, S.**The visible difference predictor: an algorithm for the assessment of image fidelity.** Digital Images and Human Vision, 1993.

6. DEITEL, H. M., DEITEL, P. J. **Java Como Programar**. Bookman, 2003.

7. GONZALEZ, R., WOODS, R. **Processamento de Imagens Digitais**. São Paulo: Editora Edgard Bluçher, primeira ed., 2000.

8. JAIN, A. **Fundamentals of Digital Image Processing**. New Jersey: Prentice-Hall Inc., primeira ed., 1989.

9. JAVA WORLD. **Program multimedia with JMF**, Disponível em: http://www.javaworld.com/javaworld/jw-04-2001/jw-0406-jmf1.html. Acessado em: 15 de Julho de 2006.

10. MARTIN, J., CROWLEY, J. **Comparison of correlation techniques**. Conference on Intelligent Autonomous Systems, Março 1995.

11. MANNOS, J. L., SAKRISON, D. J. **The effects of a visual fidelity criterion on the encoding of images.** IEEE Transactions on Information Theory, 1974.

12. MCCOWN, F**. Development of a remote object webcam controller (rowc) with corba and jmf.** Master's thesis, Donaghey College of Information Science and Systems Engineering, Arkansas, Julho 2002.

13. MULLER-SCHNEIDERS, S., JAGER, T., LOOS, H. S., NIEM, W. **Performance evaluation of a real time video surveillance system.** Proceedings 2nd Joint IEEE International Workshop on VS-PETS, Outubro 2005.

14. OLSON, E. A. **Java media framework basics.** IBM developersWorks, Maio 2002.White Paper.

15. PRATT, W. K. **Fundamentals of Digital Image Processing. California**: John Wiley & Sons, Inc., terceira ed., 2001.

16. PELI, E. **Vision Models for Target Detection and Recognition**. World Scientific Publishing, 1995.

17. PAJARES, A., ET. AL. **Jmfmod: A new system for media on demand presentation.** Proceedings of the 28th Euromicro Conference, 2002.

18. SOONG, T. T. **Fundamentals of Probability and Statistics for Engineers**. New York: John Wiley and Sons Ltd., primeira ed., 2004.

19. SPIEGEL, M. R. **Estatística.** São Paulo: Makron Books, terceira ed., 1993.

20. SUN MICROSYSTEMS, **Java Media Framework 2.0 API Guide**. Disponível em: http://java.sun.com/products/java-media/jmf/1.0/guide/index.html. Acessado em: 10 de Agosto de 2006.

21. SUN MICROSYSTEMS. **Interface QualityControl**. Disponível em: http:// java.sun.com/ products/ java-media/ jmf/2.1.1/ apidocs/javax/ media/control/ QualityControl.html. Acessado em: 7 de Setembro de 2006.

22. SUN MICROSYSTEMS. **JDesktop Integration Components (JDIC)** Disponível em: https://jdic.dev.java.net/. Acessado em: 15 de Agosto de 2006.

23. SCHULZRINNE, H., FOKUS, G., CASNER, S. **Rtp: A transport protocol for real-time applications.** in RFC1889, Internet Engineering Task Force, Janeiro 1996.

24. TRIOLA, M. F. **Introdução à Estatística**. Rio de Janeiro: Livros Técnicos e Científicos Editora, Sétima ed., 1998.

25. ZILIANI, F., CAVALLARO, A. **Image analysis for video surveillance based on spatial regularization of a statistical model-based change detection.** Swiss Federal Institute of Technology, 1997.

26. ZHOU, H., CHEN, M., WEBSTER, M. **Comparative evaluation of visualization and experimental results using image comparison metrics.** IEEE Vsualization, Outubro 2002.

27. WASTON, A. B. **Digital Images and Human Vision**. MIT Press, 1993.

28. Wilcox, J. R.. **Videoconferencing and Interactive Media: the Whole Picture**. Telecom Books, 2000.