UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

SISTEMAS DE INFORMAÇÃO – BACHARELADO

PROTÓTIPO DE UM SISTEMA DE MONITORAÇÃO UTILIZANDO CIRCUITO FECHADO DE TELEVISÃO (CFTV)

EDERSON JOSÉ

BLUMENAU
2010

2010/2-09

EDERSON JOSÉ

PROTÓTIPO DE UM SISTEMA DE MONITORAÇÃO UTILIZANDO CIRCUITO FECHADO DE TELEVISÃO (CFTV)

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Sistemas de Informação — Bacharelado.

Prof. Dalton Solano dos Reis, M.Sc. Orientador

BLUMENAU
2010

2010/2-09

PROTÓTIPO DE UM SISTEMA DE MONITORAÇÃO UTILIZANDO CIRCUITO FECHADO DE TELEVISÃO (CFTV)

Por

EDERSON JOSÉ

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Dalton Solano dos Reis, M.Sc. Orientador, FURB
Membro:	Prof. Antônio Carlos Tavares – FURB
Membro:	Prof. Miguel Alexandre Wisintainer, M.Sc. – FURB

Blumenau, 08 de dezembro de 2010.

Dedico este trabalho a toda a minha família, a minha namorada e todos os meus amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

Em primeiro lugar, a Deus, por tudo.

Aos meus pais, Francisco e Cecília, pela compreensão e apoio durante estes anos de estudo.

A minha namorada Franciele Cuco, pela compreensão e apoio durante estes anos de estudo.

Ao meu orientador, professor Dalton Solano dos Reis pela atenção dispensada a este trabalho.

Aos meus irmãos e colegas Ednilson e Odair José pelo apoio e suporte de software nas horas em que precisei.

Aos professores e colegas da FURB, pela amizade e colaboração.

A todos aqueles que, direta ou indiretamente, contribuíram para a realização deste trabalho.

A alegria está na luta, na tentativa, no sofrimento envolvido. Não na vitória propriamente dita.

Mahatma Gandhi

RESUMO

Com o acentuado crescimento da população e consequentemente da necessidade de obter segurança, o sistema de vigilância utilizando Circuito Fechado de TeleVisão tornou-se muito importante e em muitos casos imprescindível. Este trabalho apresenta um protótipo de um sistema que pode ser usado para a monitoração de ambientes utilizando uma câmera para captação de imagens a fim de registrar toda e qualquer movimentação, usando como base o TimCam (SOURCEFORGE, 2003). Aliado a isso, foi adicionado um algoritmo de comparação de imagens a fim de alcançar o objetivo principal, que era fazer um protótipo de um sistema que pudesse gerar economia de espaço físico para o armazenamento de imagens, e contendo as funções essenciais de um CFTV comercial. Utilizou-se o Java Media Framework (JMF) para a captação das imagens e a linguagem Java para seu desenvolvimento. Com a reutilização das classes do TimCam, somadas as novas funcionalidades, observou-se que num ambiente com iluminação controlada, é possível fazer uma monitoração com registros de imagens, eliminando as imagens consideradas semelhantes com uma taxa de acerto satisfatória, garantindo assim, que se deixe gravado em disco apenas o que for considerado relativo e necessário.

Palavras-chave: Sistemas de vigilância. CFTV. JMF. Java. Comparação de imagens.

ABSTRACT

With the accentuaded growth in population and consequently the need for security surveillance, a system using closed-circuit television has become very important and in many cases essential. This work presents a system prototype that can be used for monitoring environments using a camera to capture images in order to record any movement, using as the basis TimCam (SOURCFORGE, 2003). Allied to this was added an image comparison algorithm to reach the main objective, which was to create a system prototype with a resource for saving physical space for storing images, and containing the essential functions of a commercial CCTV. Was used the Java Media Framework to capture the images and language Java for development. With the reuse of classes from TimCam, added new features, it was observed that in an environment with controlled lighting, is possible to do a monitoring with images, eliminating the images considered similar with a satisfactory rate of success, and ensuring that it only saves what is considered relative and necessary.

Key-words: Surveillance systems. CCTV. JMF. Java. Image Comparison.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API – *Application Programming Interface*

AVI – *Áudio Vídeo Interleave*

BMP – *BitMap*

CODEC – Codificadores / Decodificadores

CFTV – Circuito Fechado de TeleVisão

CMY – *Cyan, Magenta and Yellow*

FTP – *File Transfer Protocol*

HSI – *Hue, Saturation, Intensity*

HTTP – *HyperText Transfer Protocol*

JMF – *Java Media Framework*

JPEG – *Joint Photographic Experts Group*

KB – *KiloByte*

MER – Modelo de Entidade e Relacionamento

MPEG – *Motion Picture Experts Group*

MSE – *Mean Square Error*

RAM – *Random Accesses Memory*

RGB – R*ed, Green, and Blue*

RTP – *Real-time Transport Protocol*

RTCP – *Real-time Transport Control Protocol*

SIP – *Session Initiate Protocol*

TCP – *Transmission Control Protocol*

UDP – *User Datagram Protocol*

URL – *Universal Resource Locator*

USB – *Universal Serial Bus*

VNC – *Virtual Network Computing*

SUMÁRIO

1 INTRODUÇÃO

Com o crescimento populacional, cresce conseqüentemente os problemas com a criminalidade como roubos, assaltos, assassinatos, entre outros. A busca por segurança é cada vez maior, seja para o uso doméstico (cuidar da casa e da família), como também institucional (empresas ou órgãos públicos). Uma pesquisa mostra já no começo da década, um grande crescimento no mercado de segurança seja por guardas ou por equipamentos específicos de prevenção.

> Para se proteger dos bandidos, indústrias, lojas e condomínios mantêm um exército de 1,3 milhões de pessoas trabalhando como seguranças em todo o país. É um contingente de guardas cujo tamanho equivale ao dobro do efetivo de toda a força policial dos 27 Estados brasileiros. O mercado de segurança vem crescendo a uma taxa de 30% ao ano. Uma em cada catorze residências brasileiras possui algum equipamento para prevenção de assalto, além das grades – o dobro do que havia cinco anos atrás (EDITORA ABRIL, 2000).

Porém na hora de procurar alguma solução em segurança, com a grande variedade de opções disponíveis em mercado, torna-se muito complicada a escolha entre o meio mais eficaz, com a melhor relação custo x benefício.

A segurança tradicional por meio de cães de guarda é facilmente driblada, tendo em vista que o animal poderá sofrer de ações imobilizadoras, perdendo sua função. A segurança por meio de vigias humanos, além de um meio caro, devido ao fator trabalhista, encargos e tributações diversas, pode ser considerada um meio não muito eficaz, perdendo sua confiabilidade em função da vulnerabilidade, uma vez que se trata de pessoas e estas estão suscetíveis a falhas, como estar ausente de seu posto, de alguma forma desacordado ou até mesmo ser rendido pelo(s) invasor(es), ficando assim apenas a versão dos fatos contada pelo guarda, sem nenhum registro visual, dificultando a busca e apreensão dos suspeitos.

O sistema de monitoramento com Circuito Fechado de TeleVisão (CFTV) auxilia de forma eficaz contra a ação indesejada de vândalos e criminosos, ficando registrada a ação, o que torna fácil a identificação dos elementos para possível tomada de providências mas, pelo fato de uma imagem ser a forma mais eficaz de reconhecer uma pessoa ou objeto, e ser uma evidência forte em um caso judicial, por exemplo, um CFTV comprova assim que é um sistema eficaz, deixando outros métodos que não venham a registrar imagens ou vídeos a desejar.

Possui uma boa relação de custo se comparado à vigilância humana, por exemplo, pois o usuário poderá comprar o sistema de acordo com a sua necessidade, residencial ou

empresarial. Sendo que este sistema pode ser utilizado como meio de segurança externo e interno, onde ele registra toda a movimentação captada pela câmera, durante 24 horas por dia, com bateria própria ligada à rede elétrica convencional, que uma vez cortada continuará funcionando normalmente de acordo com a capacidade da bateria. Desta forma, este sistema torna-se um dos mais confiáveis dentre os sistemas convencionais.

Com base nas afirmações relatadas, neste trabalho implementou-se um protótipo de um sistema de CFTV, onde por meio de uma câmera, as imagens são captadas e gravadas em um computador local, e uma cópia é enviada a um servidor remoto através de uma conexão via FTP. Assim poderão ser visualizadas e consultadas posteriormente tanto local quanto remotamente, conforme necessidade.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho foi desenvolver um protótipo de um sistema para monitoração de ambientes com CFTV que permita a captação e transmissão de imagens (fotos) utilizando como dispositivo uma câmera *webcam*, visando economia de recursos de software e de hardware.

Os objetivos específicos do trabalho são:

a) capturar imagens utilizando uma *webcam*, registrando-as com atributos temporais como, hora, dia, ano, afim de facilitar sua busca quando necessário;
b) analisar as diferenças entre as imagens registradas, para diminuir o volume de armazenamento, economizando recurso de espaço em disco;
c) ter um conjunto de funcionalidades mínimas de um CFTV, com funções essenciais para o monitoramento e acesso as imagens (visualização de imagens em tela, pesquisa de imagens já gravadas e visualização remota das imagens).

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado em quatro capítulos.

No primeiro capítulo tem-se a introdução, a justificativa do trabalho e os objetivos.

No segundo capítulo é exposta à fundamentação teórica, onde são abordados temas relevantes como CFTV, sistemas de vigilância, processamento de imagens. Além também de trabalhos correlatos.

No terceiro capítulo é abordado o desenvolvimento do trabalho. Para um melhor entendimento, este foi dividido em seções que abordam os requisitos principais, especificação, implementação, técnicas e ferramentas utilizadas e resultados.

E como quarto capítulo é exposta a conclusão final e as possíveis extensões.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentadas algumas funcionalidades de sistemas de vigilância, do circuito fechado de televisão (CFTV), do JMF que será utilizado de base para o desenvolvimento do protótipo de sistema, processamento de imagens, comparação de imagens e por fim, trabalhos correlatos.

2.1 SISTEMAS DE VIGILÂNCIA

Um sistema de vigilância tem como principal finalidade monitorar, garantir a segurança e integridade tanto de pessoas como casas, empresas e bens em geral, enfim, por qualquer coisa que se pretenda zelar. Pode ser utilizado apenas para fins de observação ou acompanhamentos, como em projetos de estudo ou projetos científicos, onde alguma determinada experiência precisa ser vigiada enquanto não se tem alguém por perto, por exemplo. Uma boa vigilância, num modo geral, não necessariamente evita que decisões erradas sejam tomadas, mas com certeza ajudam a diminuir a chance de resultados negativos que resultem em alguma perda. A vigilância pode ser feita através de várias formas, podendo ela ser feita com pessoas especializadas (vigilantes), com equipamentos eletrônicos como alarmes, com máquinas fotográficas, geralmente usada no trânsito para controle de velocidade, entre outros.

Na próxima subseção são mostrados alguns tipos de sistemas de vigilância que podem ser encontrados no mercado, assim como suas funcionalidades, pontos fracos e fortes, para que se possa entender a diferença entre eles e também para comparar as vantagens do CFTV, em relação aos demais.

2.1.1 Tipos de sistemas de vigilância

Agressivo, barulhento ou com cara de mau, o cachorro é a única defesa contra ladrões na maioria dos lares brasileiros. Eles já são 25 milhões no país, um para cada sete pessoas. Não é por acaso. O cão de guarda eficiente é muito mais que um animal que ataca invasores.

O ideal é que ele espante os intrusos antes que se animem a pular o muro. Muitas vezes, basta uma aparência que amedronte ou um latido vigoroso. Mas nem sempre o melhor amigo do homem é o melhor segurança. O cão de guarda eficiente deve saber obedecer ao proprietário, caso contrário pode tornar-se uma ameaça. Animais de raças como *rottweiler*, pastor alemão e fila tendem a ser agressivos se não tiverem tratamento correto desde pequenos (VEJA SUA SEGURANÇA, 2001).

O uso de vigilância humana assim como o animal pode sofrer influências negativas em vários fatores, como por exemplo, efeitos do sono durante a noite, principalmente, falha na vigilância por doenças ou por fadiga. Além de ser um sistema de vigilância caro, pois o vigia tem salário e uma série de direitos, que não está ao alcance da grande parte da população. Para dar uma noção de valores necessários para manter um vigilante, podemos ver no Quadro 1 os pisos salariais com vigência de março de 2009 até março de 2010 no Rio de Janeiro, onde tem-se os valores mínimos que podem ser pagos aos vigilantes, conforme seu cargo específico (Sindicato das empresas de segurança privada – Rio de Janeiro, 2009).

Função	**Salário**
Vigilante	R$ 752,80
Vigilante de escolta	R$ 978,63
Vigilante motorista/motociclista	R$ 903,84
Vigilante orgânico	R$ 752,80
Vigilante feminina/recepcionista	R$ 752,80
Agente de segurança	R$ 903,84
Agente patrimonial	R$ 903,84
Agente de segurança pessoal	R$ 903,84
Supervisor de área/Coordenador de área	R$ 1.129,82
Fiscal de posto ou supervisor de posto	R$ 833,93
Instrutor	R$ 1.267,27
Vigilante Brigadista	R$ 752,80
Vigilante condutor de cães	R$ 752,80
Vigilante responsável pelo monitoramento de aparelhos eletrônicos	R$ 752,80

Fonte: Sindicato das empresas de segurança privada – Rio de Janeiro (2010).

Quadro 1 – Piso para a categoria de vigilantes no Rio de Janeiro

É importa lembrar que além do valor do salário mostrado no quadro acima, ainda existem os encargos sociais e trabalhistas que variam de acordo com o tipo em que empresa está enquadrada. Sendo optante do simples, um salário de R$ 1.000,00 com os encargos custa R$ 1.337,80 ao contratante, e ao optante do não simples um salário de R$ 1.000,00 com os encargos custa R$ 1.681,80 ao contratante, por exemplo.

Um sistema de alarme é um conjunto de equipamentos eletrônicos que tem por finalidade informar a violação do perímetro ou local protegido, através de sinal sonoro ou visual e basicamente consiste em sensores espalhados num ambiente que detectam movimentos e emite estes sinais no local ou para alguma empresa contratada para a segurança.

A vantagem desse sistema é que pode acuar grande parte da invasão, causando medo no invasor que sai em retirada, uma desvantagem é que como o sensor detecta o movimento, se passar um animal tipo um gato ou outro qualquer, o sistema é acionado gerando um alarme falso, e uma movimentação em vão até o local monitorado, caso seja uma casa ou empresa.

2.2 CFTV

Ao se pensar em vigilância eletrônica muitos ainda têm a visão de um sistema *high tech* voltado a espionar pessoas à distância, mas a verdade é que os CFTV podem não só acrescentar segurança às residências, mas também conveniência. O sistema utilizando CFTV pode apresentar muitas vantagens em relação aos outros sistemas de vigilância. Não necessariamente como única forma de monitoramento, mas sim como uma forma mais eficaz que pode ser usada, de acordo com Digital Secury (2008), pelos seguintes motivos:

a) visualiza, monitora e grava imagens de diversos ambientes ao mesmo tempo;
b) fator psicológico de dissuasão, pois o marginal sabe que está sendo vigiado e suas imagens estão sendo gravadas;
c) inibe a ação de invasores, depredadores, pichadores e pessoas mal intencionadas;
d) manutenção barata;
e) facilita o trabalho de pronta resposta (polícia e vigilância particular) fornecendo pormenores do crime que está ocorrendo;
f) auxilia no controle de acesso de pessoas, mercadorias e veículos;
g) possível integração com sistemas de alarmes;
h) acesso às imagens pela Internet;
i) monitoramento e fiscalização dos procedimentos de segurança praticados por funcionários e moradores;
j) confiabilidade das imagens gravadas e facilidade de monitoramento remoto.

Os sistemas de CFTV digitais são mais fáceis de administrar que os sistemas

analógicos. Podendo ser integrado com instalações existentes de um CFTV ainda oferecendo acesso imediato as imagens ao vivo ou mesmo as gravadas, o armazenamento é mais simples, oferecendo um tempo de autonomia muito maior, a qualidade da imagem digital pode ser superior, além de não sofrer degradações com armazenamento.

Os sistemas digitais podem alcançar um objetivo primordial: diminuir os custos de operação resultando em um melhor custo e beneficio. Cada vez mais os benefícios do CFTV digital substituem a tecnologia anteriormente dominante, por todas as suas vantagens, mas principalmente pela possibilidade de conexão em rede, permitindo o acesso local ou remoto, redução de infra-estrutura de instalação, melhores recursos de informática, que permitem um acesso a qualquer momento e gerenciamento de permissões de acessos, gerenciamento de histórico de eventos, entre outras (GUIA DO CFTV, 2008).

Visando essas características sobre o CFTV pode-se concluir que o mesmo contém o maior número de pontos positivos em vários aspectos.

2.3 TIMCAM

O TimCam é um programa para *webcam*, escrito puramente em Java, e é de domínio publico, o código fonte não apenas livre para *download* gratuito, mas também pode ser alterado modificado ou incorporado em outros projetos (SOURCEFORGE, 2003). As suas principais características são:

a) pode usar qualquer dispositivo de *webcam*, desde que seja suportado pelo JMF;
b) tira fotos únicas;
c) tira fotos em intervalo definido em segundos;
d) envia as fotos para um site FTP.

Como requisito, é necessário que o sistema operacional utilizado, ao exemplo de Microsoft Windows, Linux, Solaris, entre outros, ofereçam suporte ao Java. É preciso ter o JMF, que é detalhado na próxima seção e uma conexão de internet, para o *upload* das imagens.

O TimCam tem como objetivo principal capturar imagens em um determinado intervalo entre cada captura, e fazer o envio dessas imagens para um site FTP especificado.

2.4 JAVA MEDIA FRAMEWORK

O JMF é uma *Application Programming Interface* (API) que permite a manipulação de áudio, vídeo e outras mídias em aplicações Java como *applets*. Com JMF é possível capturar transmissão de áudio e vídeo, e codificá-lo em diversos formatos como também a transmissão das mídias pelo padrão *Real- time Transport Protocol* (RTP) para o desenvolvimento de aplicações que utilizem vídeo sob demanda (SUN MICROSYSTEMS, 2008).

A API do JMF consiste de interfaces e classes que definem o comportamento e interação de objetos usados em aplicações de multimídia. Segundo o Java World (2009) o JMF é composto pelos seguintes componentes:

a) `datasource`: trata da conexão com a fonte da mídia, podendo ser um dispositivo de captura (microfone, câmeras digitais, por exemplo), um arquivo local ou remoto, ou um *streaming* de vídeo em tempo real disponibilizado através de uma *Universal Resource Locator* (URL). Um `datasource` encapsula o fluxo da mídia de forma parecida com a de um CD de música. No JMF, o objeto `datasource` representa o áudio, o vídeo, ou uma combinação dos dois. Um `datasource` pode ser um arquivo ou um fluxo recebido a partir da Internet. Nesta classe, depois de determinar a sua localização ou protocolo, o `datasource` incorpora tanto a mídia local, bem como o protocolo e software utilizado para fornecer os meios de comunicação;

b) `capture device`: um dispositivo de captura representa o hardware que você usa para captura de dados, como um microfone ou uma câmera de vídeo. Dados capturados podem ser integrados em um `player` para ser transmitidos, para converter os dados em outro formato, ou armazenados para uso futuro. Os dispositivos de captura podem ser categorizados como *push* ou *pull source.* Com o *pull source*, o usuário controla quando vai capturar a imagem, como exemplo, pense em uma câmera quando um usuário clica no botão para fotografar. Em contraste, um microfone funciona como um *push source* porque fornece um fluxo contínuo de dados de áudio;

c) `players`: trata do fluxo de dados do `datasource`, fazendo a demultiplexação, decodificação, a aplicação de efeitos, multiplexação e a renderização para os dispositivos de saída (fones, monitor, por exemplo). Além dos tipos de controles

(barra de posicionamento, botões, etc) para a manipulação da mídia. Um `player` tem como entrada um *stream* de áudio ou vídeo e traduz para um alto-falante ou uma tela, bem como um leitor de CD lê um CD e manda o som para os alto-falantes. Um `player` pode ter estados que existem naturalmente, porque um `player` tem de se preparar e preparar a sua fonte de dados antes que possa começar a tocar a mídia. Ele primeiro tem de procurar a faixa onde a música começa e fazer algumas outras preparações, após cerca de meio segundo dependendo do seu CD player, você começa a ouvir a música. Do mesmo modo, o `player` do JMF deve fazer certa preparação antes de poder ver ou ouvir o áudio do vídeo. Em condições normais de funcionamento, um `player` segue cada estado até que ele atinja o estado final;

d) `processors`: tem as mesmas funcionalidades do `player` e, além da possibilidade de configuração dos *codecs* e efeitos, gera outra fonte de dados para o `datasink`. Um `processor` é um tipo de `player` no JMF, a interface do `processor` se estende ao `player`, assim como o `processor` suporta os mesmos controles de apresentação, tal como o `player`, mas o `processor`, no entanto, tem controle sobre qual tratamento é realizado sobre o *stream* de entrada de mídia. O `processor` pode também converter a mídia em um `datasource` para que possa ser apresentada em outro `player` ou `processor`, ou convertida para outro formato;

e) `datasink`: os dados de mídia, proveniente de `processors`, podem ser gravados em um arquivo tornando-se outro `datasource` ou tornar uma media *stream*;

f) `format`: o objeto `format` representa o exato formato da mídia, o formato em si não carrega os parâmetros específicos da codificação ou informações, ele descreve o nome da formato de codificação e o tipo de dados que o formato exige.

Com o JMF pode-se trabalhar com os protocolos RTP e *Real-time Transport Control Protocol* (RTCP) para transmissão de mídias em redes IP. Contudo, não há suporte a mecanismos de sinalização como os oferecidos pelo H.323 e pelo *Session Initiate Protocol* (SIP). Isso não impede que algum tipo de sinalização seja feita utilizando protocolos HTTP, por exemplo. De fato, o JMF trabalha com o perfil RTP, para conferências com pouco ou nenhum controle (GOUVEIA, 2008).

Alguns formatos de arquivos conhecidos são o H.261 e H.263 para vídeo; G.723 para áudio. A comunicação envolvendo JMF pode ser compreendida, basicamente, em três

estágios, a captura o processamento e transmissão das imagens.

2.4.1 Transmissão das imagens

As imagens coletadas pela câmera ligada ao sistema são gravadas em disco rígido. Para o disco a imagem vem das câmeras conectadas através de portas *Universal Serial Bus* (USB). As imagens por sua vez, também serão enviadas a um computador remoto ligado por uma conexão de rede.

Para a transmissão de dados multimídia em tempo real na internet, protocolos de comunicação devem ser devidamente escolhidos a fim de atender da melhor forma possível os requisitos desse tipo de comunicação, tendo em vista que os dados transmitidos em tempo real possuem como característica a necessidade de atrasos constantes. Para as comunicações multimídia, os protocolos de aplicação normalmente são focados em dois aspectos: dados (áudios, vídeo e dados convencionais) e controle. O tratamento dos dados multimídia, para que possam ser transmitidos por rede, é feito por algoritmos chamados *codecs*. Já o controle das comunicações pode ser executado por arquiteturas publicas de comunicação multimídia, como o H.323, por exemplo.

O *Transmission Control Protocol* (TCP) é ainda o protocolo mais utilizado na internet, as aplicações que necessitam de transmissões confiáveis foram projetadas para usufruir dos recursos oferecidos por esse protocolo. Entretanto, as características de operação do TCP o tornam inadequado para aplicações de transmissão em tempo real na internet. Outro fator que também deve ser considerado nessas transmissões é a variação de atraso (*jitter*). Como as comunicações trabalham com mídias continuas como, áudio e vídeo, variações no atraso podem ocasionar perdas de continuidade na reprodução dessas mídias.

Outro fator que prejudica o uso do protocolo TCP em ambientes multimídia, é que esse protocolo não suporta comunicações *multicast*, essa tecnologia de nível de rede permite a economia de recursos de comunicação quando as mesmas informações são transmitidas de uma fonte para mais receptores. Para que as transmissões *multicast* sejam possíveis, protocolos não orientados a conexão devem ser utilizados, o que desabilita o TCP de operar nesse cenário.

Por fim, o protocolo *User Datagram Protocol* (UDP), esse protocolo não oferece qualquer mecanismo de controle de erro ou de fluxo, não sendo orientado a conexão. Assim o protocolo UDP atende as necessidades das aplicações de comunicação multimídia em tempo

real. O UDP é uma das soluções mais usadas para transmissão de dados de mídia em tempo real na internet (GOUVEIA, 2008).

2.4.1.1 Protocolo RTP

O protocolo RTP conceitualmente provê transporte fim-a-fim necessário a aplicações de tempo real na internet. Mais especificamente, esse protocolo tem como objetivo fornecer um mecanismo para levar dados sensíveis ao atraso, por exemplo, áudio e vídeo, de uma extremidade a outra na rede em tempo real. Em relação a esse aspecto de transporte, o RTP atua como um intermediador entre os dados a serem transmitidos e os meios que efetivam a transmissão.

O protocolo RTP foi designado para ser independente das camadas de rede e de transporte. Sendo independente o RTP pode ser implementado sobre qualquer protocolo, embora seu uso seja mais comum sobre o UDP. Com a utilização de UDP as transmissões multimídia se beneficiam com a simplicidade do protocolo, como a ausência de controle de erro e de fluxo. O RTP foi primeiramente desenvolvido para utilização em conferências multimídias com muitos participantes, porém outros cenários de comunicação são possíveis. Funcionalmente o RTP opera tanto em redes *unicast* quanto em redes *multicast*, desde que esteja encapsulado sobre o UDP (GOUVEIA, 2008).

2.5 PROCESSAMENTO DE IMAGENS

Para trabalhar com imagens capturadas pelas câmeras é necessário o processamento das imagens, isso para encontrar, por exemplo, um formato que não ocupe tanto espaço e mantenha uma boa qualidade. Processar uma imagem consiste em transformá-la sucessivamente, utilizando técnicas para filtrar objetos e descartar dados irrelevantes. De acordo com Gonzalez e Woods (2000, p. 1) o interesse desses métodos decorre de duas áreas: melhoria de informação visual para a interpretação humana e o processamento de dados de cenas para percepção automática através de máquinas.

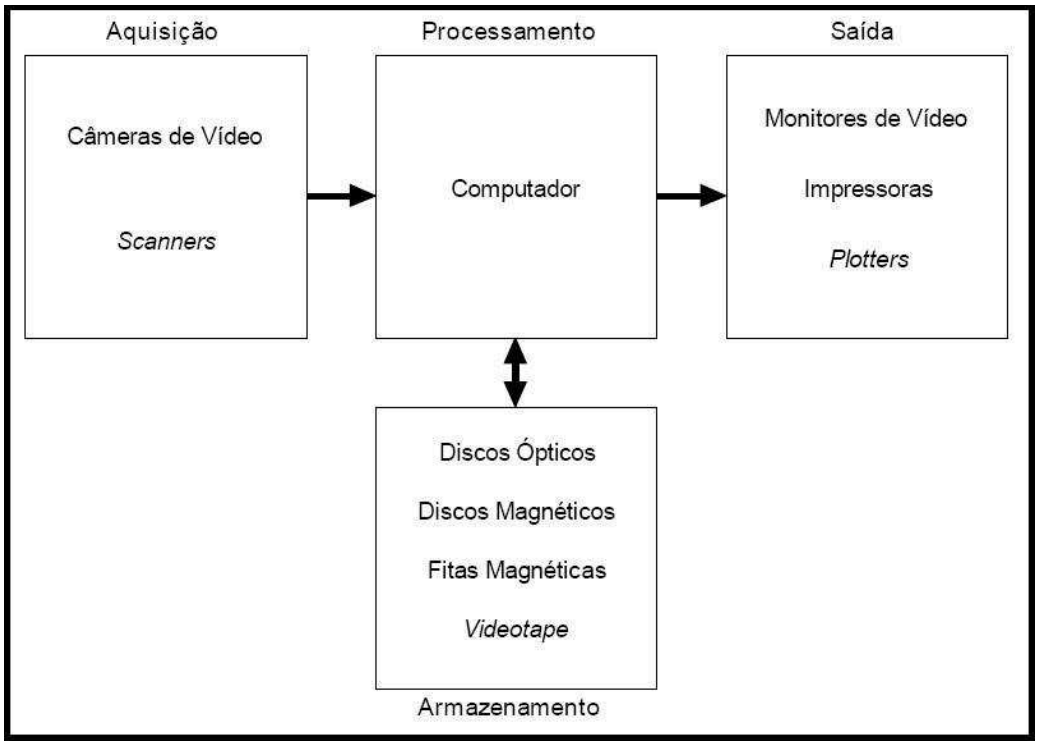
Outro fator importante a ser considerado é o formato em que as imagens serão gravadas, já que trabalhar com imagens multimídia requer uma escolha cuidadosa de um

formato adequado para os arquivos de imagem, e as seguintes características em relação ao formato dos arquivos são importantes: número de cores suportadas, resoluções, grau de compressão, entre outros (PAULA FILHO, 2000).

Segundo Paula Filho (2000, p. 20), os métodos de compressão de vídeo permitem que seu usuário defina uma série de parâmetros de qualidade, e um sistema pode trabalhar com vários algoritmos diferentes de compressão, dependendo da qualidade desejada e da resolução. Esses algoritmos são chamados de COdificadores-DECodificadores (CODECS). Para concluir, alguns dos formatos de vídeo mais comuns são *Motion Picture Experts Group* (MPEG), MPEG-1, *Audio Video Interleave* (AVI), entre outros.

2.5.1 Processamento de imagens dividido em componentes

De acordo com Marques Filho e Vieira Neto (1999) os sistemas desde os de baixo custo até as sofisticadas aplicações que envolvem um complexo uso de imagens pode ser representado pelo diagrama conforme a Figura 1. Ele abrange as principais operações que se pode obter sobre uma imagem, que são a aquisição, o armazenamento, o processamento e a exibição. Uma imagem pode também ser transmitida a distância utilizando meios de comunicação como internet, por exemplo.



Fonte: Marques Filho e Vieira Neto (1999, p. 24).

Figura 1 – Elementos no sistema de processamento de imagens

A etapa de aquisição e digitalização de imagens tem como função converter uma imagem monocromática, por exemplo, em uma representação numérica adequada para o processamento digital subseqüente. O bloco compreende dois elementos principais, o primeiro é um dispositivo físico e segundo o digitalizador propriamente dito, que converte sinais elétricos e analógicos em informação digital, ou seja, que pode ser representada por 0s e 1s, segundo Marques Filho e Vieira Neto (1999, p. 24).

Uma imagem digital é uma imagem $f(x,y)$, portanto uma imagem digital pode ser vista como uma matriz cujas linhas e colunas identificam um ponto na imagem, cujo valor corresponde ao nível de cinza naquele ponto. Uma imagem monocromática pode ser descrita matematicamente pela função $f(x,y)$ da intensidade luminosa, sendo seu valor em qualquer ponto de coordenadas espaciais (x,y), proporcional ao brilho da imagem naquele ponto.

Segundo Marques Filho e Vieira Neto (1999, p. 24) a função $f(x,y)$ representa o produto da interação entre iluminância $i(x,y)$ que exprime a quantidade de luz que incide sobre o objeto e as propriedades de refletância ou transmitância próprias do objeto, que podem ser representadas pela função $r(x,y)$ cujo valor exprime a fração de luz incidente que o objeto vai transmitir ao ponto (x,y). Matematicamente: $f(x,y) = i(x,y).\ r(x,y)$ com: $0 < i\ (x,y) < \infty$ e $0 < r(x,y) < 1$.

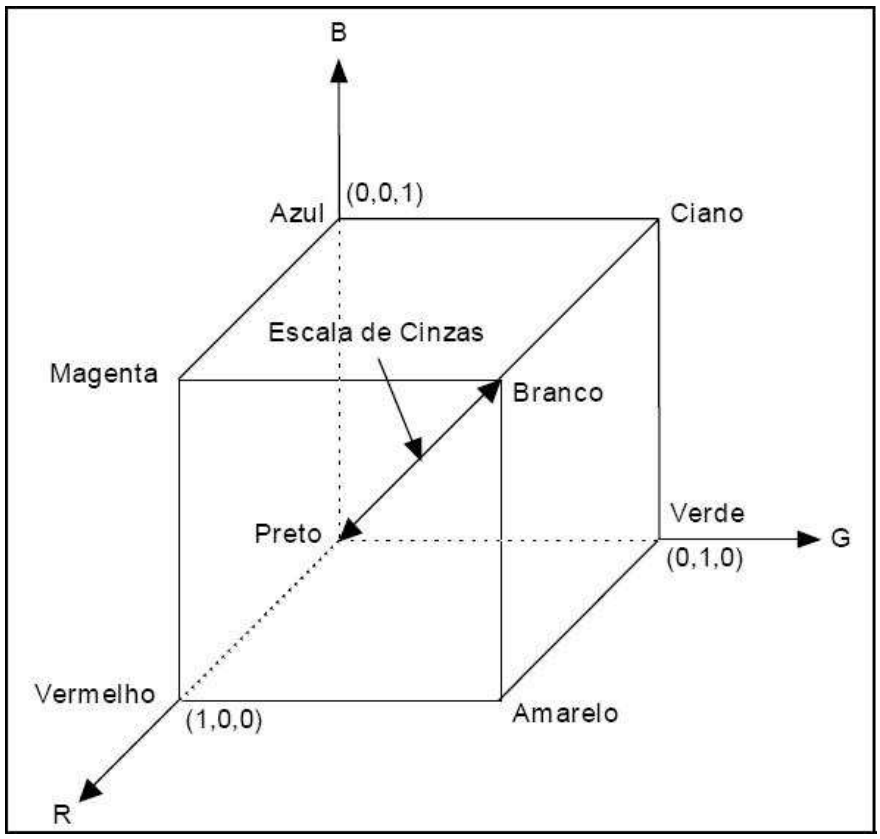
A teoria da percepção cromática pelo olho humano baseia-se em hipótese formulada por Young em 1801, em que os cones, células fotossensíveis que compõem a retina juntamente com os bastonetes, subdividem-se em três classes com diferentes máximos de sensibilidade situados em torno do Red, Green and Blue (RGB). Estas três cores são denominadas cores primárias aditivas, pois é possível obter qualquer outra cor a partir de uma combinação aditiva de uma ou mais delas, em diferentes proporções.

O objetivo dos modelos de cores é permitir a especificação de cores em um formato padronizado e aceito por todos. Em linhas gerais, um modelo de cores é uma representação tridimensional na qual cada cor é representada por um ponto no sistema de coordenadas 3-D. Também são muito utilizados como modelos de representação de cores os padrões Cyan, Magenta, Yellow (CMY), Hue, Saturation, Intensity (HSI).

O YCbCr é formado da seguinte forma, o Y significa a iluminância ou intensidade, Cb seria a crôminancia azul, ou mais precisamente, o desvio da cor cinza em um eixo azul-amarelo, finalmente Cr, que seria a crôminancia vermelha ou, o desvio da cor cinza em um eixo vermelho-ciano. O verde pode ser calculado com base nestes três valores. Na maioria das codificações de cor RGB tem uma amostra diferente de R, G e B, o mesmo não acontece com as codificações de YCbCr, essas variantes operam na evidência empírica que o olho humano é

mais sensível as variações de intensidades do pixel do que as variações das cores propriamente ditas. Assim cada pixel possui associados uma amostra de Y e grupos que de pixels que compartilham amostras de Cb e Cr (MULTIMEDIAWIKI, 2009).

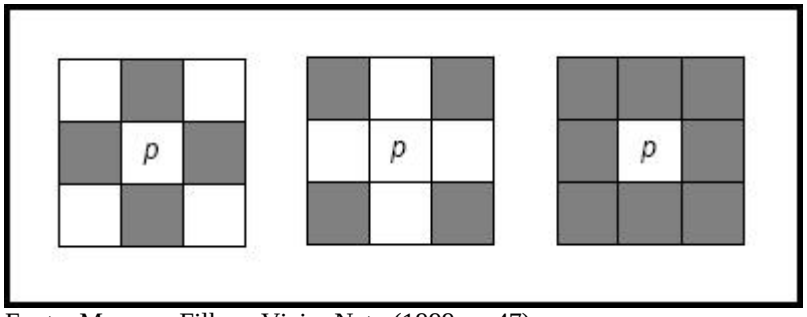
O modelo RGB é baseado em um sistema de coordenadas cartesianas, que pode ser visto como um cubo onde três de seus vértices são as cores primárias, outros três as cores secundárias, o vértice junto à origem é o preto e o mais afastado da origem corresponde à cor branca, conforme ilustra a Figura 2. Neste modelo, a escala de cinza se estende através de uma linha (a diagonal do cubo) que sai da origem (preto) até o vértice mais distante dela (branco). Por conveniência, geralmente assume-se que os valores máximos de R, G e B estão normalizados na faixa de 0 a 1. O modelo RGB é o mais utilizado por câmeras e monitores de vídeo, segundo Marques Filho e Vieira Neto (1999, p. 121).



Fonte: Marques Filho e Vieira Neto (1999, p. 121).

Figura 2 - Modelo RGB

Um pixel p, de coordenadas (x,y) tem 4 vizinhos horizontais e verticais cujas coordenadas são (x+1,y), (x-1,y), (x,y+1) e (x,y-1). Estes pixels formam a chamada 4-vizinhança de p que é designada por N4(p). Os quatro vizinhos diagonais de p são os pixels de coordenadas (x-1,y-1), (x-1,y+1), (x+1,y-1) e (x+1,y+1), que constituem o conjunto Nd(p). A 8-vizinhança de p é definida como N8(p) = N4(p) U Nd(p). Os tipos de vizinhança são ilustrados na figura 3.



Fonte: Marques Filho e Vieira Neto (1999, p. 47).

Figura 3 – Conceitos de 4-vizinhança, vizinhança diagonal e 8-vizinhança

A conectividade entre pixels é um importante conceito usado para estabelecer limites de objetos e componentes de regiões de uma imagem e para se estabelecer se dois pixels estão conectados. Então se faz necessário determinar se eles são adjacentes segundo algum critério e se seus níveis de cinza satisfazem a um determinado critério de similaridade, por exemplo, em uma imagem binária onde os pixels podem assumir valores 0 e 1, dois pixels podem ser 4-vizinhos mas, somente serão considerados 4-conectados se possuírem o mesmo valor.

Um pixel p é adjacente a um pixel q se eles forem conectados. Há tantos critérios de adjacência quanto são os critérios de conectividade, dois subconjuntos de imagens, S1 e S2, são adjacentes se algum pixel em S1 é adjacente a algum pixel de S2, sendo S um conjunto de pixels.

Um caminho (*path*) de um pixel p de coordenadas (x,y) a um pixel q de coordenadas (s,t) é uma seqüência de pixels distintos de coordenadas: (x0, y0), (x1, y1), ... , (xn, yn), onde: (x0, y0) = (x,y) (xn, yn) = (s,t)(xi, yi) é adjacente a (xi-1, yi-1)1 ≤ i ≤ n, onde n é denominado o comprimento do caminho.

Um grande desafio num projeto de sistemas de processamento de imagens é o armazenamento, isso em virtude da grande quantidade de bytes necessários para tanto, conforme Marques Filho e Vieira Neto (1999, p. 24), este armazenamento pode ser dividido em três categorias: (1) armazenamento de curta duração de imagem, enquanto ela é utilizada nas várias etapas do processamento, normalmente é o papel da memória *Random Access Memory* (RAM), (2) armazenamento de massa para operações de recuperação de imagens relativamente rápidas, requer uso de discos magnéticos e (3) armazenamento de imagens para recuperação futura, quando necessário.

Com exceção das etapas de aquisição e exibição, a parte do processamento das imagens normalmente é expressa em forma de algoritmos, e a maioria das funções pode ser implementada via software. Segundo Marques Filho e Vieira Neto (1999, p. 25), a tendência do mercado de hardware é a comercialização de placas genéricas compatíveis com os padrões

de barramento consagrados pelas arquiteturas populares de microcomputadores. O software de controle dessas placas é que determina sua aplicação específica a cada situação, as vantagens mais imediatas são as reduções de custo, modularidade, independência de fornecedor.

O monitor de vídeo é um elemento fundamental no sistema, a tecnologia ainda muito usada é tubo de raios catódicos (TRC), mas o número de monitores de cristal liquido, comum em *notebooks* vem ficando cada vez mais popular.

As imagens captadas e digitalizadas podem ser transmitidas para qualquer lugar utilizando rede de computadores e protocolos de comunicação já existentes. O desafio da transmissão de imagens é a quantidade de bytes que precisa ser transferida de um local a outro, tem-se a dependência de fatores como velocidade de banda, por exemplo. Este problema ainda é mais sério quando se pretende transmitir seqüências de vídeo em tempo real, onde fatores como sincronização, por exemplo, deve ser considerado.

2.6 COMPARAÇÃO DE IMAGENS

O presente trabalho tem como uma das principais finalidades ser econômico de recursos, como espaço em disco, por exemplo, sendo assim é necessário achar uma forma de comparar as imagens capturadas a fim de eliminar as que são semelhantes. Para avaliar fidelidade entre imagens e analisar o grau de semelhança entre duas imagens pode se usar um modo de comparação pixel a pixel pelo motivo de que já existem algoritmos que fazem esse cálculo, portanto, podem ser adaptados para o protótipo de sistema do CFTV. Pode-se ainda, dividir os métodos quanto à forma de análise das matrizes correspondentes às imagens, com métodos de análise pixel a pixel, abaixo na seção 2.6.1 é mostrado o método pixel a pixel do erro médio quadrático e o da distância Euclidiana.

2.6.1 Métodos de análise pixel a pixel

Essa área é composta por métodos que se utilizam essencialmente da análise matemática sobre os valores das matrizes de cores das imagens. Tais métodos são comumente chamados de distâncias. A baixa complexidade e custo computacional as tornam ainda

bastante utilizadas, embora exista um número significativo de estudos que demonstram a sua ineficiência para análise de qualidade visual (ALBUQUERQUE, 2008).

Um exemplo de método é a distância euclidiana, que conforme equação mostrada abaixo na Figura 4, representa em valor absoluto a diferença entre o valor original e o valor de teste.

$$D_{euclidiana} = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

Fonte: Albuquerque (2008).

Figura 4 – Equação Euclidiana

Outro método de comparação pixel a pixel é usando o erro médio quadrático, do inglês M*ean Square Error* (MSE). Apesar de conhecida e bastante utilizada para análise de fidelidade, essa métrica não apresenta bons resultados em diversos casos. É possível analisar imagens modificadas por diferentes filtros de distorção e com o mesmo valor de MSE (ALBUQUERQUE, 2008). O cálculo da equação do MSE é dado pela equação conforme Figura 5, e quanto menor seu valor absoluto menor o erro. Onde `m` e `n` são as dimensões da matriz, `I` é a imagem alvo e `I'` é a imagem de referência.

$$MSE = \frac{1}{m * n} \sum_{y=1}^{m} \sum_{x=1}^{n} \left[I(x,y) - I'(x,y) \right]$$

Fonte: Albuquerque (2008).

Figura 5 – Equação do erro médio quadrático

2.7 TRABALHOS CORRELATOS

Com relação a trabalhos correlatos, são descritos sistemas que apresentam algumas das funcionalidades que serão apresentadas no protótipo de sistema de CFTV desenvolvido. Neste contexto, o sistema Gvi (TELEALARME, 2008) foi escolhido algumas de suas funções, as quais foram implementadas no protótipo desenvolvido. A demonstração deste sistema

comercial tem como objetivo mostrar o que está disponível no mercado e também suas características.

Na tela principal do sistema (Figura 6) encontram-se os painéis com as funções do Gvi, encontra-se a visualização das câmeras com sua nomenclatura de identificação, um quadro com data e hora, um painel com botões onde podem ser selecionadas todas as câmeras ligadas ao sistema, para que essas fiquem visíveis ou habilitadas para configurações. Logo abaixo deste painel tem um outro quadro com as opções de visualização das câmeras, sendo que estas podem ser alteradas conforme preferência do usuário, quando selecionado o botão da procura inteligente, é aberta uma tela onde se pode pesquisar por arquivos já gravados anteriormente ao que está sendo gravado no momento, permitindo que se abra o arquivo e se visualize.

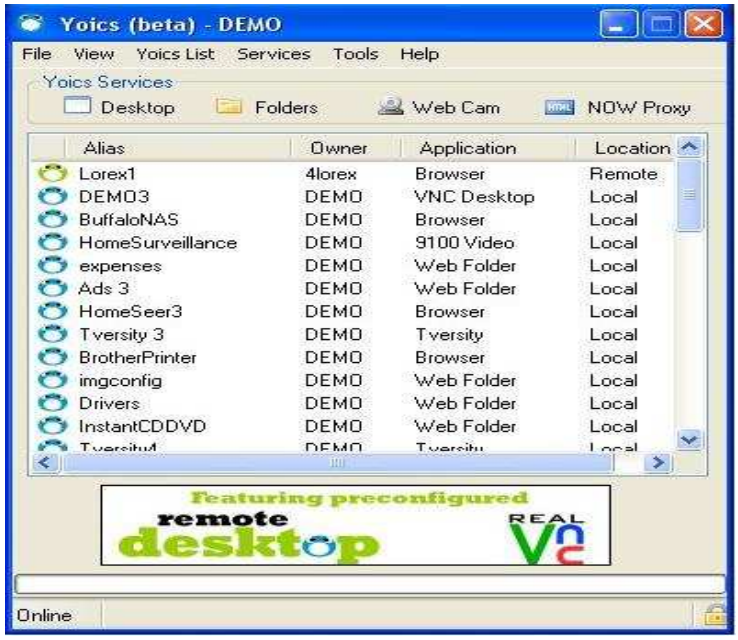
Conforme exibido na Figura 6, encontra-se outro botão de configurações, onde se altera local onde são salvos arquivos, alterações de vídeo como, cores e zoom. Ainda tem uma procura rápida, que consiste em manipular o arquivo que está sendo gravado, podendo ser retrocedido ou passado para frente. O controle de acesso do sistema é para que apenas usuários cadastrados possam fazer as alterações. Para isto, primeiro entram com um código de usuário e senha e depois tem acesso para mexer no que for preciso. E finalmente, o botão de fechar, para a saída do sistema. Desta forma temos a seguinte legenda: nome de cada câmera (1), painel data e hora (2), seleção de câmeras (3), modos de exibição (4), procura inteligente (5), configurações (6), procura rápida (7), login (8), sair do sistema (9).

Fonte: TeleAlarme (2008).

Figura 6 – Interface do sistema de segurança GVI

O Yoics é uma alternativa para acessar remotamente computadores, arquivos, *webcams*, ip cameras, tendo uma série de ferramentas disponíveis, que formam um ótimo sistema de segurança. O Yoics é instalado na máquina e é uma solução para uso em rede que transforma qualquer computador ou dispositivo de rede em um recurso facilmente acessível e compartilhável pela Internet. Caracterizado por uma intuitiva interface de usuário, acesso remoto e compartilhamento, relativamente simples como usar mensagens instantâneas.

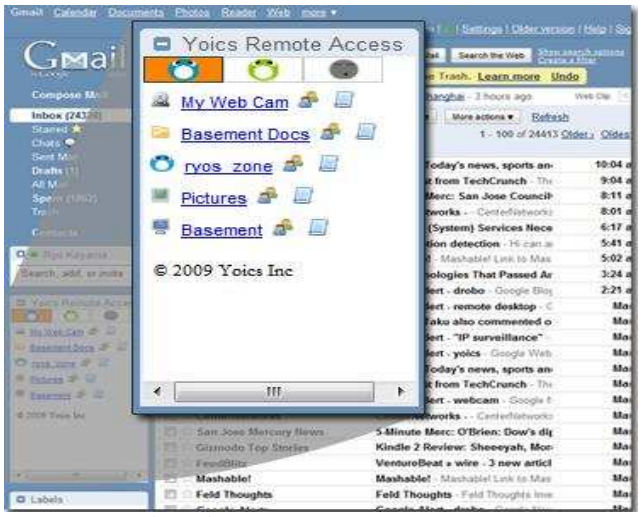
O desktop do Yoics se assemelha e funciona como um programa de mensagens instantâneas, o Yoics tem uma lista com vários dispositivos, esses dispositivos podem ser seus, ou que foram compartilhados por você. Este por sua vez, pode ser tornado acessível via rede em qualquer lugar. Durante a instalação pode-se optar por algumas funções da ferramenta, como por exemplo, acesso remoto via *Virtual Network Computing* (VNC), pasta para compartilhamento, compartilhamento de fotos ou finalmente de uma *webcam*. Uma vez configurado a pasta do Yoics, ela se torna instantaneamente acessível e compartilhável na internet, via *browser* ou *webDav*, que é uma extensão do protocolo HTTP para transferência de arquivos. Na figura 7 abaixo, tem-se a tela do Yoics, onde são listados os dispositivos.



Fonte: Yoics (2008).

Figura 7 – Interface do sistema Yoics

Depois de instalado o Yoics é necessário criar um login e senha para poder acessar o sistema, pois ele pode ser acessado de qualquer lugar, inclusive tem disponível um *gadget* que pode ser adicionado no Gmail. Uma vez instalado, é possível acessar as *webcam*s que já estão configuradas ou demais dispositivos em seu servidor, conforme Figura 8 abaixo.



Fonte: Yoics (2008).

Figura 8 – *Gadget* do Yoics que pode ser adicionado ao Gmail

3 DESENVOLVIMENTO

Neste capítulo são abordadas as etapas de desenvolvimento do protótipo de CFTV, ilustrando os principais requisitos, especificação, técnicas da implementação e por fim resultados e discussão.

3.1 REQUISITOS PRINCIPAIS DO CFTV

O protótipo de sistema de segurança com CFTV deverá:

a) permitir usuário visualizar imagem da câmera na interface (Requisito Funcional – (RF));
b) permitir configurar dados do servidor remoto via FTP onde serão enviadas as imagens já capturadas (RF);
c) permitir configurar intervalo de captura entre uma imagem e a outra (RF);
d) exibir na tela detalhes da conexão via FTP com servidor (RF);
e) exibir na tela status das imagens sendo gravadas e enviadas (RF).
f) comparar cada imagem nova que é capturada com a imagem anterior a fim de eliminar a última, caso sejam semelhantes (RF);
g) permitir usuário pesquisar por imagens registradas, já gravadas em banco de dados (RF);
h) deverá utilizar câmeras IP para captação de imagens (Requisito Não Funcional – (RNF));
i) deverá ser desenvolvido na linguagem Java utilizando o NetBeans (RNF).

3.2 ESPECIFICAÇÃO

O protótipo apresentado utiliza alguns dos diagramas da *Unified Modeling Language* (UML). Para os diagramas de caso de uso, de classe e de seqüência foi utilizada a ferramenta StarUML versão 5.0.2.

3.2.1 Diagrama de casos de uso

A Figura 9 apresenta o diagrama de casos de uso contendo as principais interações do usuário com sistema.

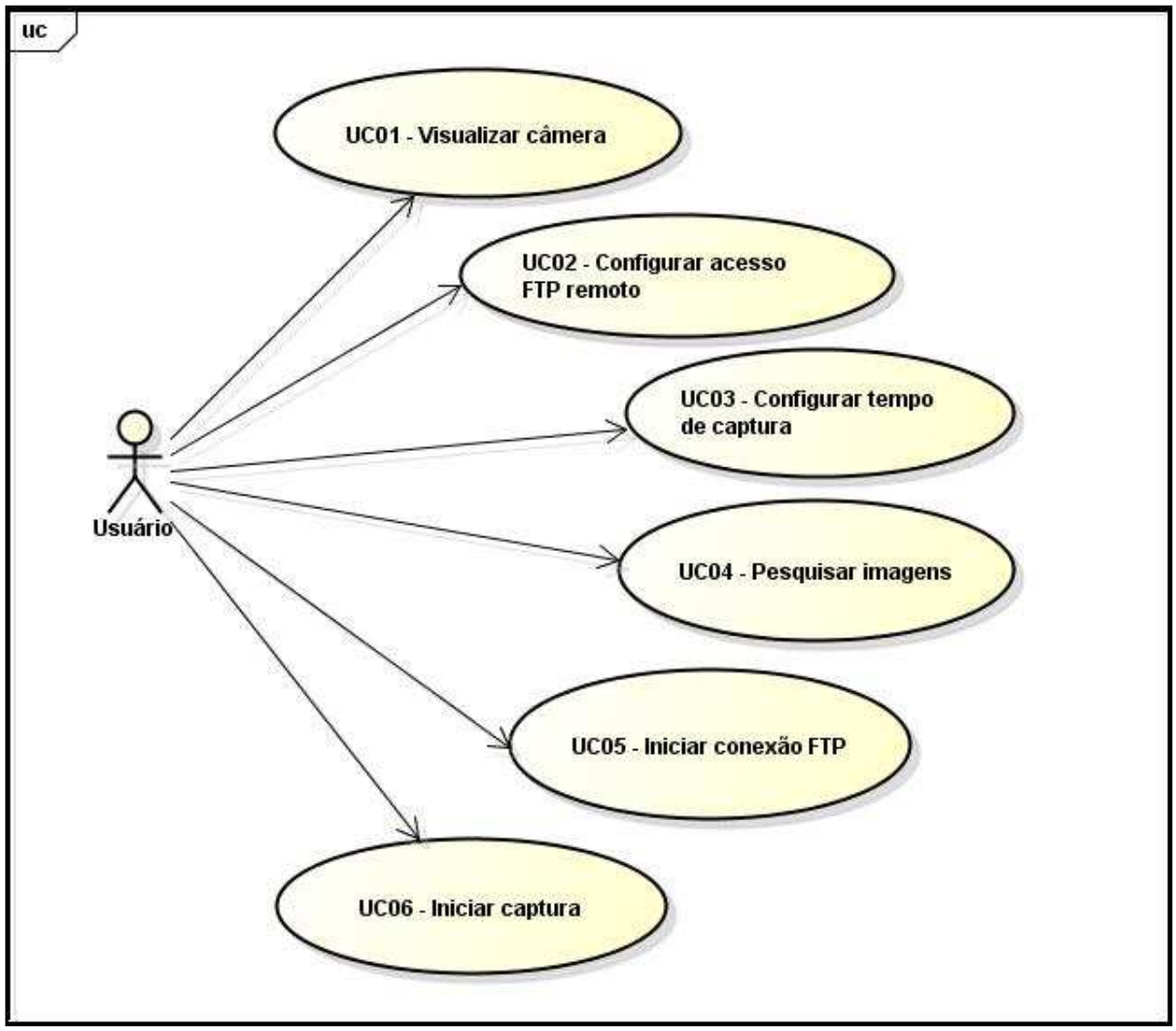


Figura 9 – Diagrama de casos de Uso

O caso de uso `Visualizar câmera` (Quadro 2) descreve que o usuário pode visualizar a imagem da câmera sendo via USB ou própria do computador, no caso de um notebook, por exemplo. Este caso de uso possui um cenário principal, um alternativo e um de exceção.

Visualizar câmera: possibilita ao usuário visualizar na tela a imagem da câmera.	
Pré-condição	Um dispositivo de captura precisa estar instalado.
Cenário principal	1) O protótipo lista os dispositivos encontrados. 2) usuário seleciona dispositivo. 3) O protótipo valida o dispositivo.
Fluxo alternativo 01	No passo 1, caso tenha apenas um dispositivo instalado, o protótipo automaticamente o seleciona.
Exceção 01	No passo 1 caso não tenha um dispositivo de captura instalado, o protótipo gera erro e não executa.
Pós-condição	O dispositivo é definido com sucesso.

Quadro 2 – Caso de uso visualizar imagens

O segundo caso de uso `Configurar acesso FTP remoto` (Quadro 3), explica como o usuário faz para configurar o acesso a um servidor remoto, utilizando um acesso via FTP.

Configurar acesso FTP remoto: permite ao usuário inserir dados de acesso a um servidor remoto, onde serão enviadas imagens obtidas pela *webcam*.	
Pré-condição	Protótipo aberto com dispositivo detectado. Ter um servidor FTP remoto, com conta e senha disponível.
Cenário principal	1) Usuário preenche campos com dados da conta FTP, sendo: nome de usuário, senha, endereço e porta.
Exceção	No passo 1 do cenário principal, caso seja informado algum dado errado, na tela onde é exibido detalhes da conexão será exibido o erro ocorrido.
Pós-condição	Protótipo pronto para conexão com servidor via FTP.

Quadro 3 – Caso de uso Configurar acesso FTP remoto

No terceiro caso de uso `Configurar tempo para captura` (Quadro 4) é explicado como o usuário pode definir o tempo de captura entre uma imagem outra, sendo este sempre em segundos.

Configurar tempo para captura: permite ao usuário definir um tempo em segundos, em o que o dispositivo vai levar entre uma captura e a seguinte.	
Pré-condição	Protótipo aberto com dispositivo detectado.
Cenário principal	1) Usuário digita um tempo em segundos, para definir tempo entre cada captura de imagem.
Pós-condição	Protótipo capturando imagens, guardando em disco e enviando ao servidor FTP no tempo digitado.

Quadro 4 – Caso de uso Configurar tempo para captura

No quarto caso de uso `Pesquisar imagens` (Quadro 5) é explanado como será a pesquisa por imagens do arquivo, registradas no dia ou anteriormente.

Pesquisar imagens: Permite ao usuário fazer busca por imagens em arquivo, a fim de consultas.	
Pré-condição	Ter imagens capturadas e gravadas.
Cenário principal	1) Usuário seleciona data para pesquisa por imagem. 2) Usuário seleciona faixa do horário para consulta. 3) Usuário clica OK e pesquisa por data. 4) Usuário visualiza imagens na tela do protótipo.
Cenário Alternativo 01	No passo 3 do cenário principal, o usuário pode clicar no nome de uma imagem e abri-la no editor do MS Windows ao invés de ver no próprio protótipo.
Pós-condição	Protótipo traz imagens gravadas na data selecionada na pesquisa.

Quadro 5 – Caso de uso Pesquisar imagens

No quinto caso de uso `Iniciar conexão FTP` (Quadro 6) é mostrado como o usuário faz para iniciar a conexão com o servidor remoto, a partir do momento que ele já tenha previamente configurado os dados para conexão na tela, conforme mostrado no caso de uso UC02.

Iniciar conexão FTP: permite ao usuário iniciar a conexão via FTP.	
Pré-condição	Protótipo aberto com dispositivo detectado. Ter os dados para conexão já digitados na tela de configuração.
Cenário principal	1) Usuário pressiona botão iniciar FTP.
Cenário Alternativo 01	No passo 1 do cenário principal, o usuário pode também interromper a conexão clicando em Parar FTP.
Pós-condição	Protótipo conectado e pronto para envio das imagens.

Quadro 6 – Caso de uso Iniciar conexão FTP

Finalmente, no sexto caso de uso `Iniciar captura` (Quadro 7) é explicado como o usuário faz para iniciar a captura das imagens.

Iniciar captura: permite ao usuário iniciar a captura das imagens.	
Pré-condição	Protótipo aberto com dispositivo detectado. Ter os dados para conexão já digitados na tela de configuração.
Cenário principal	1) Usuário pressiona botão Iniciar captura.
Cenário Alternativo 01	No passo 1 do cenário principal, o usuário pode também interromper a captura clicando em Parar captura.
Pós-condição	Protótipo captando imagens, gravando em banco e enviando para servidor remoto.

Quadro 7 – Caso de uso Iniciar conexão FTP

3.2.2 Diagrama de classes

O diagrama de classes apresentado na Figura 10 mostra uma visão de como os pacotes e as classes estão estruturadas. No diagrama estão as classes mais relevantes para o entendimento do protótipo. Nesta seção são descritas como as classes se relacionam entre si para o funcionamento do sistema

.

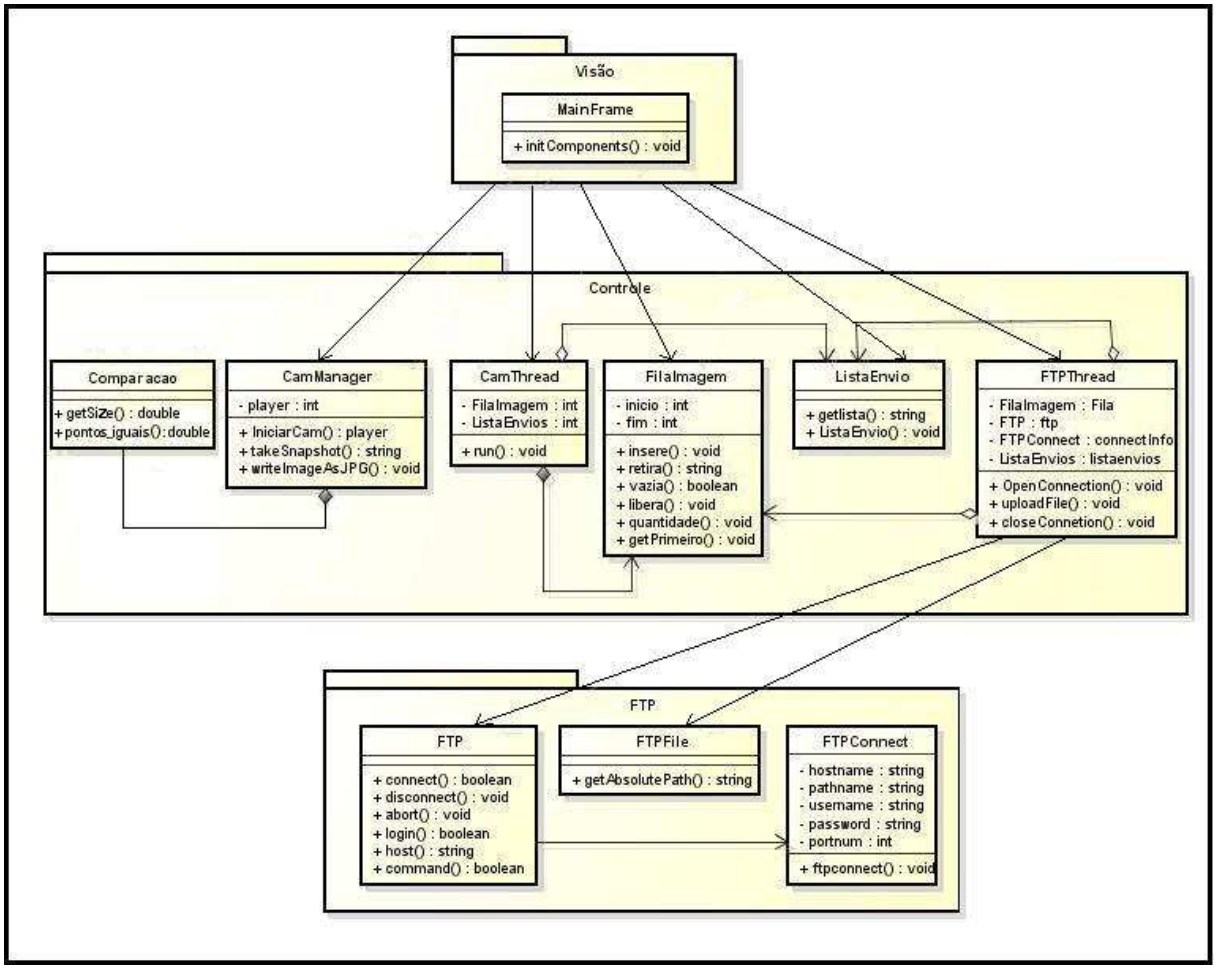


Figura 10 – Diagrama de classes

A especificação do trabalho está dividida em três pacotes: `Visão`, `Controle` e `ftp`. A classe do pacote `Visão` realiza chamadas para as classes do pacote `controle` que por sua vez realiza chamadas para as classes do pacote `ftp`. O retorno das execuções é realizado de forma inversa, do pacote `ftp` para `Controle` e pacote `Controle` para `Visão`.

O pacote `Visão` é o pacote responsável pela interação com o usuário e pela captura dos eventos da tela. Nele estão contidas as classes `MainFrame`.

A classe `MainFrame` é uma classe que armazena todos os componentes e suas propriedades, e possui a função de capturar os eventos da tela da ferramenta realizando o gerenciamento entre as demais classes.

O pacote `Controle` é responsável por prestar suporte aos demais pacotes da ferramenta. A classe `CamManager` é responsável por reconhecer o dispositivo de câmera e receber a imagem da mesma. O protótipo não permite ao usuário alterar configurações de visualização, ficando a configuração padrão da câmera como a utilizada. Ainda no pacote `Controle` encontra-se a classe `Comparação` que é responsável por analisar a similaridade entre as imagens que vão sendo captadas, e elimina as consideradas iguais dentro da

tolerância parametrizada.

A classe `Camthread` é utilizada para capturar a foto de tempo em tempo, isso em intervalo de segundos, sendo esse configurado pelo usuário. Na classe `FilaImagem` as imagens são armazenadas, após os arquivos são inseridos na classe `ListaEnvio` de onde são enviadas via FTP a um servidor definido pelo usuário, através da classe `FtpThread`.

O pacote `ftp` possui as classes `FTP`, `FtpFile` e `FtpConnect`, o objetivo desse pacote é fazer uma conexão utilizando o protocolo FTP, por onde serão enviadas as imagens já capturadas. O pacote `ftp` é executado através da classe `FtpThread` do pacote `controle`.

3.2.3 Diagrama de seqüência

A Figura 11 mostra o diagrama de seqüência, que permite visualizar o caso de uso `Iniciar captura` de uma forma reduzida para melhor visualização. A partir do momento em que o usuário iniciou o protótipo, a classe `MainFrame` inicia o processo `IniciaCam` que tem por objetivo iniciar o dispositivo com as configurações padrões da câmera. Em seguida o usuário clica no botão `Iniciar captura`, onde inicializa a captura das imagens através da classe `CamThread`. Após serem capturadas no método `TakeSnapshot` as imagens são comparadas, pela classe `Comparação` e caso sejam diferentes serão enviadas via FTP e também salvas no banco de dados. As imagens quando capturadas tem seu nome definido por hora, minuto, dia, mês, ano e número seqüencial, esse nome será utilizado para consultas posteriores e visa facilitar identificação de quando foram registradas. Esse processo de iniciar captura é realizado de forma seqüencial e é repetido de tempos em tempos, definido pelo usuário. O usuário poderá interromper o processo através do botão `Parar Captura`.

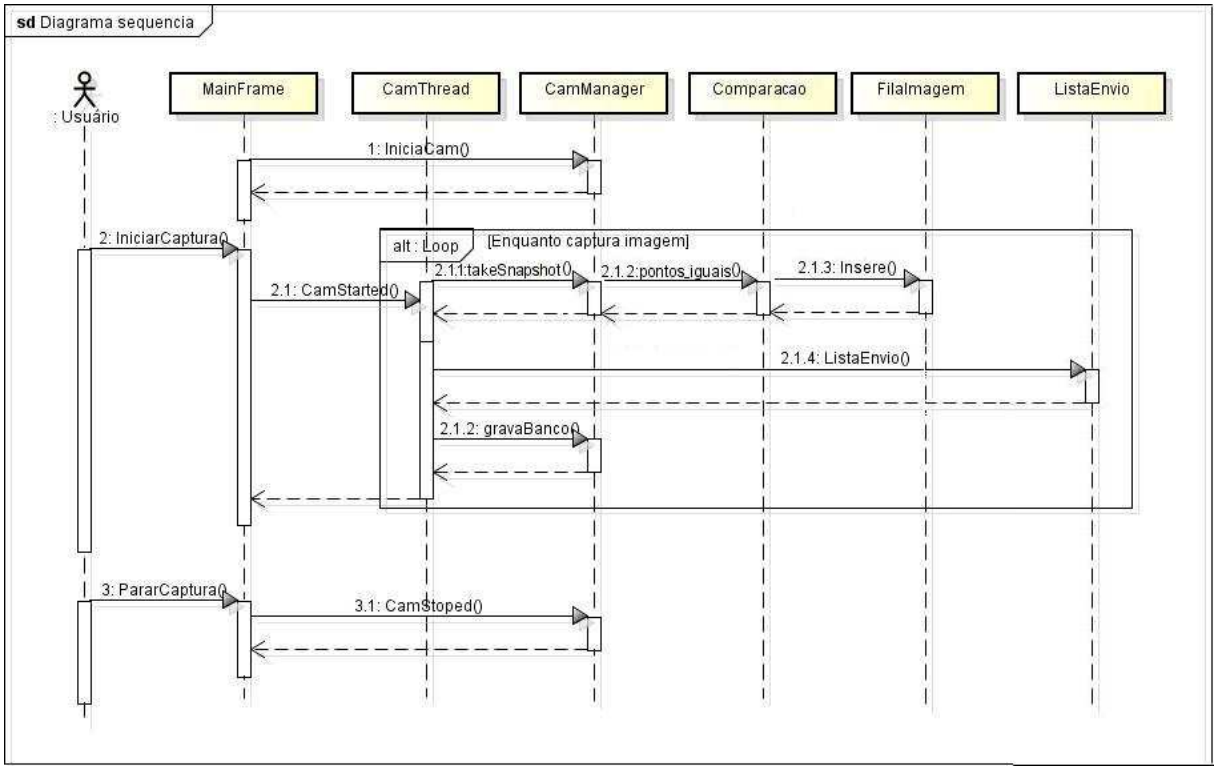


Figura 11 – Diagrama de seqüência do caso de uso `Iniciar captura`

3.2.4 Armazenamento em banco de dados

Para arquivar e controlar as imagens de forma mais segura ao ambiente, foi utilizado um banco de dados. Como banco de dados foi utilizado o MySql sever 5.1 e o MySql Query Browser versão 1.2.17 para criação e manipulação da tabela utilizada. A estrutura é simples, pois foi necessária a criação de apenas duas tabelas, uma chamada `images` e outra chamada `usuário`. Na tabela `images é` onde as imagens são gravadas com um atributo chamado `id`, que é a chave primaria, esse campo recebe uma numeração seqüencial para cada novo registro. O atributo `nome` onde fica gravado o nome da imagem, outro atributo chamado `foto`, do tipo `blob`, que armazena a imagem convertida em binário pelo codec `JPEGImageEncoder`, em um *array* de bytes.

Por fim, `DiaMes` que como diz o nome guarda a data em que o registro foi inserido no banco de dados, a consulta feita pelo usuário no protótipo é feita com uma busca pelos valores deste campo. Na tabela `usuário` é registrado o usuário de acesso e também a senha. Na Figura 12 abaixo, pode-se visualizar estrutura do banco pelo Modelo de Entidade e Relacionamento (MER).

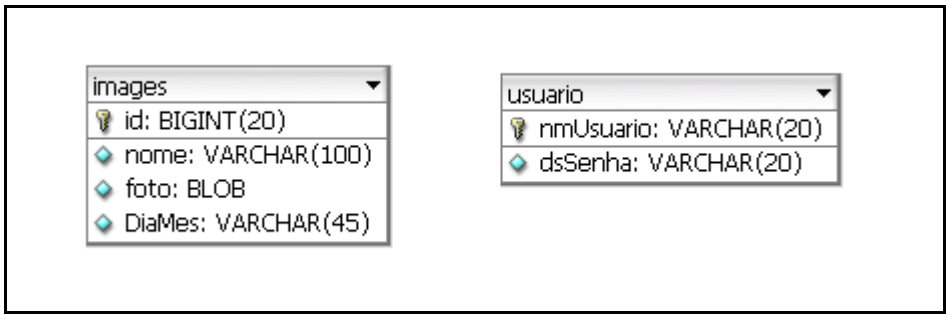


Figura 12 – MER exemplificando estrutura do banco de dados

3.3 IMPLEMENTAÇÃO

Nesta seção são abordadas as etapas do desenvolvimento do projeto. São mostradas as técnicas e ferramentas utilizadas, a operacionalidade da implementação e por fim são listados os resultados e discussões e conclusão.

3.3.1 Técnicas e ferramentas utilizadas

Para a implementação do protótipo foi utilizada a linguagem de programação Java com o ambiente de desenvolvimento Netbeans IDE, versão 6.5.1, juntamente com Java Media Framework API (JMF). Foram utilizadas classes do TIMCAM (SOURCEFORGE, 2003) para a parte de captura de imagens e envio via FTP. Como banco de dados foi utilizado o MySql sever 5.1 e o MySql Query Browser versão 1.2.17 para criação e manipulação da tabela utilizada.

3.3.1.1 Captura imagem

A captura da imagem é realizada através da classe `CamThread`, pela função `run` conforme mostrado no Quadro 8, o protótipo começa a capturar a partir da *webcam* e armazená-las localmente. Após as imagens serem capturadas, o protótipo monta o nome do arquivo, salvando localmente e as insere na classe `FilaImagem`. Posteriormente, a imagem será utilizada na classe `ListaEnvio` que será transmitida para o servidor remoto. Este

processo é realizado continuamente até que seja interrompido.

```
public void run() {
    while (MainFrame.camStarted) {
        try {
            this.arquivoLocal = CamManager.takeSnapshot(snapshotFilename);
            if (!this.arquivoLocal.equals("")) {
                this.fila.insere(arquivoLocal);
                this.listaenvios.listaEnvio(arquivoLocal.substring(0, arquivoLocal.indexOf(";")), 0);
            }
            sleep(delayInterval * 1000);
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }
}
```

Quadro 8 – Trecho de código da classe `CamThread` responsável pela captura

3.3.1.2 Transmissão das imagens ao servidor - FTP

Na classe `FtpThread` se inicializará o processo de conexão e transmissão das imagens capturadas para o servidor remoto.

Com os parâmetros definidos pelo usuário (endereço servidor, caminho, porta, usuário e senha) a classe executa a função `OpenConection` que realizará a conexão com o servidor remoto. Após a conexão estabelecida é executada o método `UploadFile` que verifica a imagem gravada na classe `FilaImagem`, transmitindo-a para o servidor destino. Esse processo executará enquanto existir imagens na fila para serem transmitidas. Após esse processo a conexão é finalizada, conforme se pode ver respectivamente no Quadro 9 e no Quadro 10 abaixo.

```
public class FtpThread extends Thread {
    public FtpThread(MainFrame mainframe, FilaImagem fila,ListaEnvios listaenvios,
            String hostname, String port, String username, String password,
            String ftpFilename, boolean onceOnly) {
        ftp = new Ftp();
        this.mainframe = mainframe;
        this.listaenvios = listaenvios;
        this.fila = fila;
        this.ftpFilename = ftpFilename;
        simpleFileUpload = onceOnly;
        connectInfo = FtpConnect.newConnect("ftp://" + hostname + ":" + port);
        connectInfo.setUserName(username);
        connectInfo.setPassWord(password);
        nmArq=ftpFilename;
       extensao = ".jpg";
    }
    private void openConnection() throws IOException {
        ftp.connect(connectInfo);
    }
    public void run() {
        try {
            openConnection();
             if (simpleFileUpload) {
                 uploadFile(snapshotFilename);
                } else {
                while (mainframe.ftpStarted) {
                   this.arquivoLocal = fila.retira();
                    uploadFile(arquivoLocal);
                    mainframe.compactaarq.liberaCompacta();
                 }
            }
            closeConnection();
        } catch (Throwable t) {
            System.err.println("Exception: " + t.getMessage());
            t.printStackTrace();
        }
    }
```

Quadro 9 – Classe `FtpThread` responsável pelo envio das imagens

No trecho de código do Quadro 10 a alteração feita na classe `FtpThread` mudou a forma de envio, na forma original do TimCam a imagem era gravada no computador, no diretório que era definido na configuração feita pelo usuário e depois enviado para o servidor de FTP. Como os arquivos agora estão armazenados em um banco de dados, a imagem agora primeiramente é buscada através de um *select* e depois então enviada via FTP.

```
private void uploadFile(String filename) {
    try {
        uploadCounter += 1;
        if (uploadCounter == 5) {
            uploadCounter = 0;
            mainframe.txtArea.setText("");
        }
        String dado[] = filename.split(";");
        ftpFilename=nmArq+dado[0];
        this.listaenvios.listaEnvio(ftpFilename,1);
        FtpFile tmpFile = new FtpFile(ftpFilename + ".tmp", ftp);
        //conexão  com o banco
          Connection conexao = getConnection();
          PreparedStatement stmt = (PreparedStatement)
        conexao.prepareStatement("select * from images  where NOME= ?");
        stmt.setString(1,dado[0]);
        ResultSet rs = stmt.executeQuery();
        while(rs.next())
        {
         InputStream input=  rs.getBinaryStream("FOTO");
         ByteArrayOutputStream output = new ByteArrayOutputStream();
          byte[] rb = new byte[1024];
          int ch = 0;
         while ((ch = input.read(rb)) != -1){
            output.write(rb, 0, ch); }
        byte[] b = output.toByteArray();
         ByteArrayInputStream bais = new ByteArrayInputStream(b);
         ImageIcon image= new ImageIcon(b);
         Image im=image.getImage();
         writeImageAsJPG(im, "C:\\"+dado[0]);
        }
        filename="C:\\"+filename;
        DataOutputStream outstream = new DataOutputStream(new FtpOutputStream(tmpFile));
        DataInputStream instream = new DataInputStream
                  (new FileInputStream(filename.substring(0,filename.indexOf(";"))));
        ...
       FtpFile destFile = new FtpFile(ftpFilename, ftp);
       tmpFile.renameTo(destFile);
    } catch (Throwable t) {
        System.err.println("Exception: " + t.getMessage());
        t.printStackTrace();
private void closeConnection() {
    ftp.disconnect();
```

Quadro 10 – Método `UploadFile` da classe `FtpThread` responsável pelo envio das imagens

3.3.1.3 Comparação de imagens

Como um método para redução de recursos físicos, no caso, espaço em disco, foi implementada uma comparação de imagens a fim de eliminar as imagens que não tenham sofrido quase ou nenhuma alteração. A classe `CamManager` que faz a chamada do método `imagem_similar`, esse método inicia a comparação passando as duas imagens que são analisadas, sendo sempre a penúltima e última que foram tiradas para a classe criada chamada `Comparação`, que por sua vez, faz o cálculo do nível do tolerância e traz o resultado, e só então a imagem é gravada em banco e enviada ao servidor remoto, caso tenham sido

consideradas semelhantes, apenas a primeira imagem capturada fica gravada, a outra é eliminada.

No método `imagem_similar` as imagens que vão ser comparadas são enviadas para a classe `Comparação`, nessa classe as imagens são medidas para ver se tem a mesma dimensão, sendo de mesmo tamanho, as imagens entram num `for`, que percorre as posições `x` e `y` das imagens pegando o valor do RGB através do `getRGB`, o detalhamento do funcionamento é explanado de forma detalhada na seção de resultados e discussões.

Abaixo no Quadro 11 está o código da classe `Comparação` onde é feita a comparação das imagens.

```
public class Comparacao {
    private int tolerancia = 95;
    public Comparacao() {
    }
    public void setTolerancia(int tolerancia) {
        this.tolerancia = tolerancia;

    }
    public double getSize(BufferedImage img_1) {
        return img_1.getHeight() * img_1.getWidth();
    }
    public double pontos_iguais(BufferedImage img_1, BufferedImage img_2) {
        long t0 = System.currentTimeMillis() , t1;
        double pontos = 0, igualdade = 0;
        // Se for de outro tamanho , retorna -1
        if (img_1.getWidth() != img_2.getWidth() || img_1.getHeight() != img_2.getHeight()) {
            return -1;
        }
        for (int x = 0; x < img_1.getWidth(); x++) {
            for (int y = 0; y < img_1.getHeight(); y++) {
                double pont_1 = img_1.getRGB(x, y) * -1,
                       pont_2 = img_2.getRGB(x, y) * -1;
                if (pont_1 > pont_2) {
                    igualdade = (pont_2 / pont_1) * 100;
                } else
                if (pont_2 > pont_1){
                    igualdade = (pont_1 / pont_2) * 100;
                }
                else {
                    igualdade = 100;
                }
                // Verifica o quanto o pixel é similiar
                if (igualdade >= tolerancia) {
                    pontos++;
                }
            } // Fim Imagem

        }
        t1 = System.currentTimeMillis() - t0;
        return pontos / getSize(img 1) * 100;
```

Quadro 11 – Classe Comparação

3.3.2 Operacionalidade da implementação

Ao executar o aplicativo é apresentada a tela de login do protótipo. Nela o usuário precisa entrar com um código de usuário e senha válidos, já cadastrados previamente no banco de dados (Figura 13), tanto para o código de usuário e senha caso sejam digitados de forma incorreta, é exibida uma mensagem ao usuário alertando do erro, a forma definida na implementação, exige que se diferenciem letras minúsculas das maiúsculas.

Figura 13 – Tela de login do protótipo de CFTV

Depois de o acesso ser validado, é apresentada a tela principal do protótipo. Através desta tela são disponibilizadas todas as opções do protótipo, que se encontram divididas por função específica, mas todas concentradas na mesma janela (Figura 14).

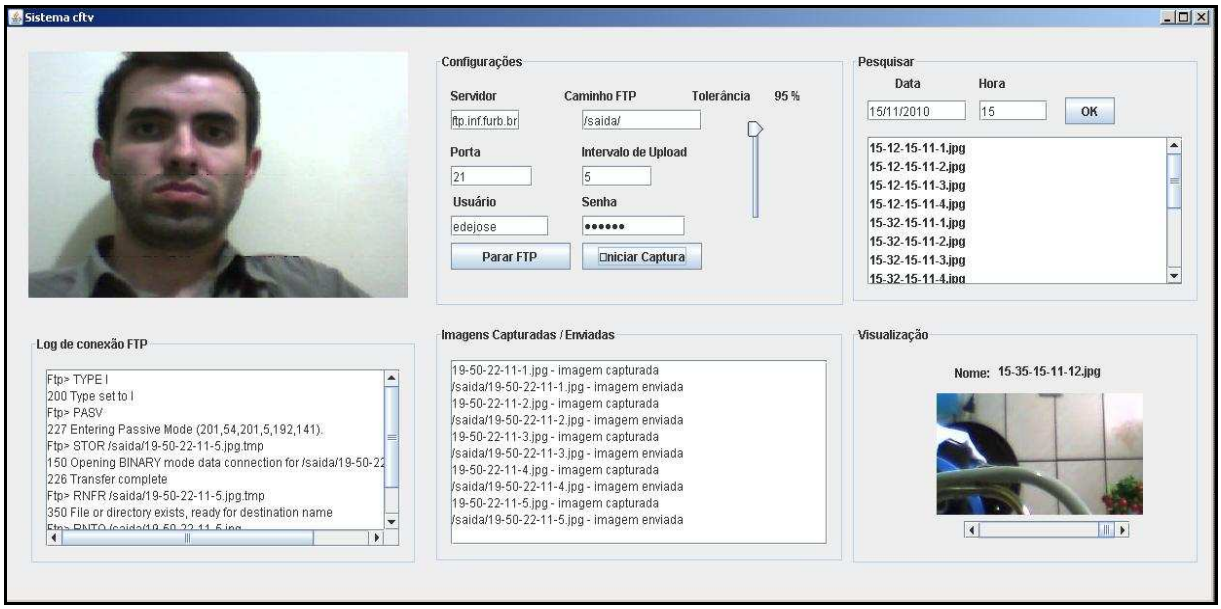


Figura 14 - Tela principal do protótipo

Nas próximas seções, é descrita uma seqüência lógica de utilização do protótipo, através da apresentação das telas, separadas por partes, para guiar e facilitar a utilização de cada função contida no protótipo.

3.3.2.1 Configurando o protótipo

Como o objetivo do protótipo é segurança, as imagens são salvas no banco de dados do servidor do sistema e também enviadas por FTP para um servidor remoto.

A configuração precisa ser feita para os seguintes campos:

a) servidor FTP: o usuário precisa colocar o nome de um servidor que tenha configurado e liberado para gravar imagens que serão enviadas;
b) caminho do FTP: dever ser digitado a pasta ou diretório no servidor para onde a imagens serão enviadas;
c) porta: o número da porta utilizada pelo protocolo de transmissão;
d) intervalo de captura: esse é o tempo em segundos, definido pelo usuário para ser o intervalo em que as imagens serão capturadas e enviadas;
e) usuário: digitar o nome de usuário de acesso ao servidor, um que seja válido e liberado para gravação de arquivos;
f) senha: senha definida do usuário para o acesso ao servidor de FTP.

Esta configuração pode ser vista na tela do protótipo conforme a Figura 15.

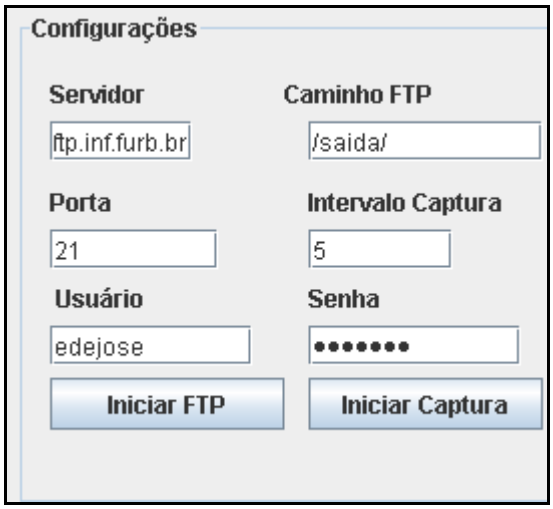


Figura 15 - Tela da configuração do protótipo

3.3.2.2 Iniciar e finalizar FTP

Uma vez completada as configurações, conforme mostra acima a Figura 15, o usuário pressiona o botão `Iniciar FTP`, o qual ao ser acionado inicializará o processo de conexão ao servidor remoto. O processo de conexão pode ser acompanhado pela parte da tela que mostra o *status* da conexão conforme é mostrado na Figura 16 abaixo.

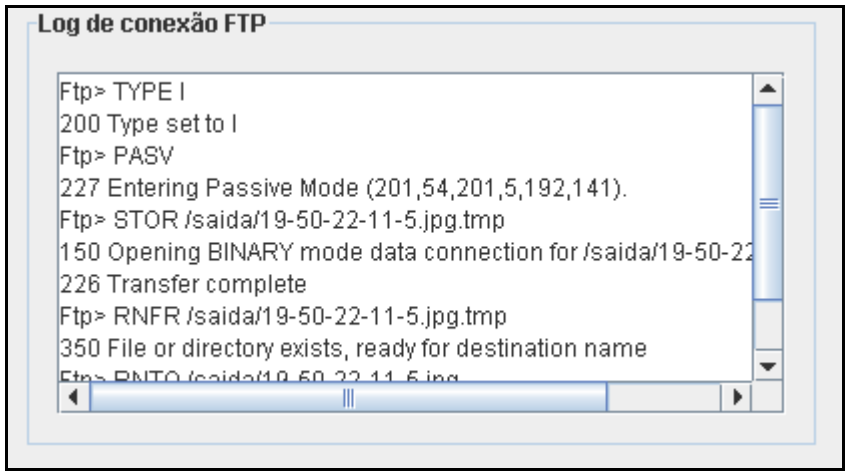


Figura 16 – Tela do protótipo com *status* da conexão FTP

O FTP pode ser inicializado a qualquer momento enquanto o protótipo captura as imagens, assim que conectado, se a captura estiver inicializada, o protótipo começa a transmitir as imagens para o servidor FTP. Dá mesma forma a conexão com o FTP pode ser finalizada a qualquer momento e com isso o envio das imagens é interrompido, mas continua sendo gravado localmente.

Caso o FTP seja reconectado o envio das imagens continua de onde parou no momento

da desconexão. Após o usuário clicar no botão `Iniciar FTP` a descrição do botão muda para `Parar Captura`, para que a conexão termine.

3.3.2.3 Iniciar e finalizar captura

A tela a seguir apresenta no protótipo onde é dado o comando para o inicio da captura e para o fim da captura, que é feito respectivamente através do clique do mouse no botão `Iniciar Captura` e `Parar Captura`.

Após o usuário clicar no botão `Iniciar Captura` o protótipo começa a capturar e armazenar as imagens localmente e a descrição do botão muda para `Parar Captura`. A partir desse momento o usuário pode parar quando desejar. Nessa mesma tela também se pode ver o *status* da captura e do envio, conforme Figura 17.

Nessa tela ainda é mostrado o nome da imagem capturada, sendo esse montado com hora, minuto, dia, mês e também um número seqüencial, na mesma tela é mostrado o *status* do envio, mostrando o caminho no servidor e a mensagem imagem enviada, o que pode parar caso a conexão seja interrompida. Nesse caso continua sendo mostrado apenas que a imagem foi capturada.

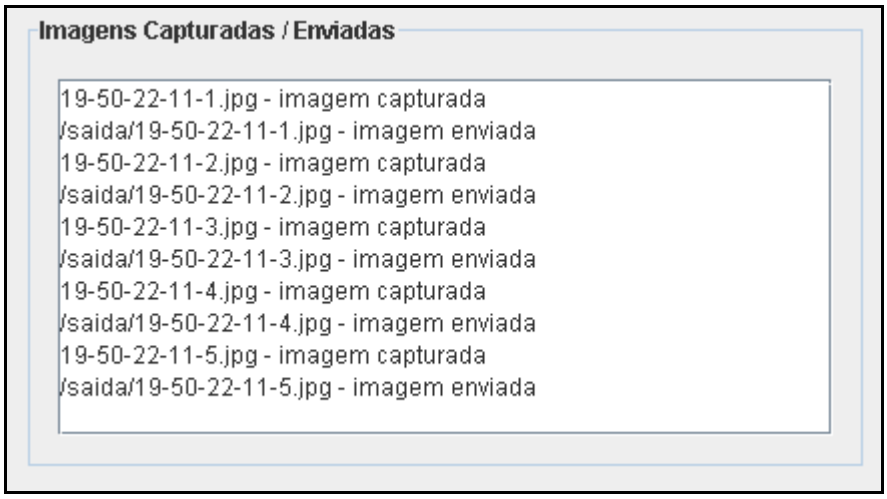


Figura 17 – Tela Parar captura e exibição de *status*

As imagens armazenadas localmente serão enviadas automaticamente para o servidor remoto se o FTP estiver conectado, caso o servidor seja conectado posteriormente as imagens serão todas transmitidas desde o momento do início da captura.

3.3.2.4 Pesquisar Imagens

Sendo o objetivo do protótipo a monitoração, torna-se essencial ter uma forma de consulta das imagens obtidas pela câmera. A pesquisa das imagens poderá ser realizada pelo próprio protótipo, onde as imagens são armazenadas em um banco de dados. As imagens serão pesquisadas depois de armazenadas, para que mostrem fatos já ocorridos, como por exemplo, identificar responsáveis por um furto.

Para pesquisar o usuário digita a data que desejar sendo essa composta por dia, mês e ano, e também a faixa do horário desejado, o protótipo recupera as imagens feitas no dia, todos os registros efetuados nesse período serão mostrados na tela, como é demonstrado na Figura 18. O usuário pode percorrer as imagens que foram selecionadas e estão listadas na tela e através de um clique ele visualiza a imagem em tamanho grande no editor padrão de imagens do MS-Windows.

Depois de escolhida a data da pesquisa, faixa de hora e pressionado o botão `OK`, a tela da pesquisa vai exibir a listagem das imagens conforme é mostrado na Figura 18.

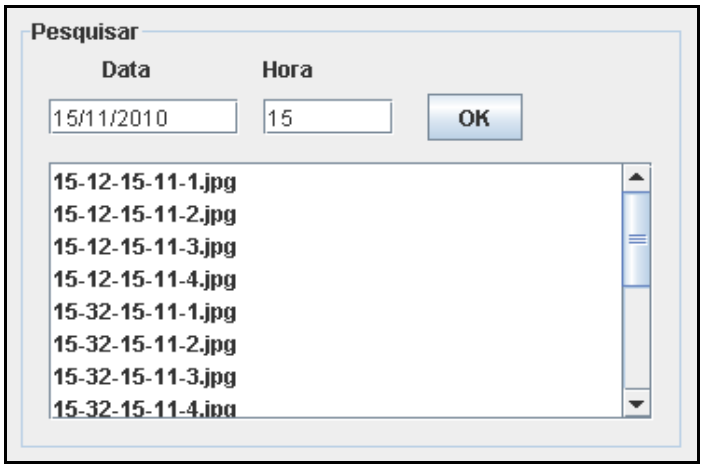


Figura 18 – Tela para pesquisar imagens com resultado de uma consulta

Depois da tela preenchida com o resultado da consulta e a imagem desejada selecionada com um clique do mouse, ela é aberta no editor conforme pode ser visto na Figura 19, onde uma janela era monitorada.

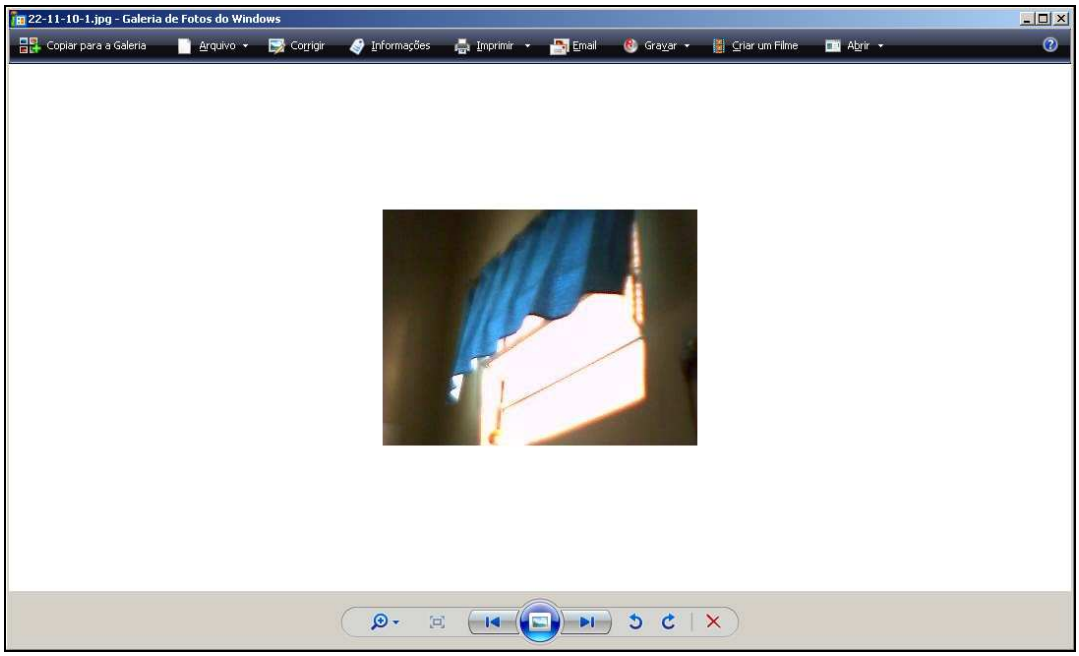


Figura 19 – Visualização de imagem no editor do MS-Windows

Outra forma de visualizar as imagens pode ser feita na própria tela do protótipo, quando o usuário faz a consulta conforme mostrado na Figura 18, a primeira imagem da lista, caso tenha registros, já aparece na tela conforme a Figura 20 abaixo. Esta é uma forma mais rápida de visualização, pois nesse caso o usuário não precisa abrir cada imagem separadamente através de um editor do sistema operacional, ele pode visualizá-las no protótipo.

As imagens são exibidas em miniaturas, e usando o botão de navegação que se encontra logo abaixo da imagem o usuário pode pular de uma para outra, clicando na seta para ir para a seguinte ou na seta para a anterior. Como podem ter muitos registros, o usuário pode rolar a barra de rolagem e passar mais rapidamente a seleção, podendo ir da primeira direto para a última.

Acima da imagem fica o nome dela, sendo esse constituído por ano/mês/dia/hora/minuto/seqüencial, então ao mudar de uma imagem para outra sempre aparece a informação com o nome dela, facilitando a localização caso precise ser resgatada do banco de dados.

Figura 20 – Visualização da pesquisa na tela do protótipo

As imagens enviadas para o servidor remoto via FTP não serão visualizadas na tela pelo do protótipo, elas são copiadas para uma pasta que foi definida no caminho conforme Figura 15 e caso o usuário queira acessá-las remotamente, poderá faze-lo via *browser* de internet ou pelo *Windows Explorer* no caso do MS-Windows, por exemplo. Neste último pode ser feito digitando o endereço completo do caminho onde estão as imagens, no caso deste trabalho, o caminho usado foi `ftp.inf.furb.br/saida`, em seguida é necessário validar o acesso com um usuário e senha.

Por último, o valor de percentual de tolerância também é alterado na interface do software pelo usuário. Ao abrir o protótipo o valor inicial da tolerância é 95%, valor que foi utilizado nos testes e aqui é tido como base.

Deslizando o botão do slide (Figura 21) o usuário pode definir o valor da tolerância, indo de 0 a 100, sendo que 0 significa que todas imagens serão consideradas semelhantes e excluídas, por ter total tolerância, ou 100 onde todas são consideradas diferentes e serão gravadas.

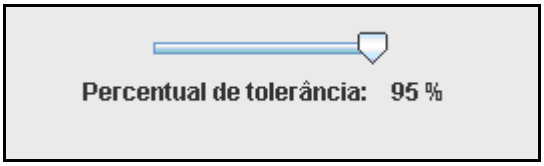


Figura 21 – Configuração do percentual de tolerância

3.4 RESULTADOS E DISCUSSÃO

O protótipo desenvolvido tem como função a monitoração de ambientes através de

uma *webcam.* O requisito essencial a ser alcançado foi o de rodar em computadores com poucos recursos de hardware, visando especialmente a redução de espaço em disco, para isso optou-se por utilizar um algoritmo que compara imagens e as grava somente quando necessário, com base nos métodos da distância euclidiana e o do erro médio quadrático. Essa comparação consiste em eliminar as imagens que forem consideradas semelhantes, de acordo com as definições da implementação e que são analisadas nesta seção.

A definição do que é considerado aqui como funções essenciais de um CFTV foi obtida com base numa entrevista feita com um usuário de um CFTV comercial, chamado Jean Carlos de Souza (SOUZA, 2010) que o utiliza em seu trabalho, na empresa CECRED. Ao ser questionado sobre qual era a principal dificuldade em utilizar o sistema que possuía, a resposta foi direta: dificuldade na consulta das imagens gravadas.

Segundo Jean, uma consulta chegou a levar mais de duas horas, devido à demora do sistema ao resgatar o que era procurado na enorme quantidade de imagens gravadas em disco, além de serem arquivos com grande tamanho. Por muitas vezes ocorreram travamentos em que o sistema precisou ser reiniciado, o que geralmente lhe acarreta em muito tempo envolvido apenas para uma simples consulta. Além dessa característica, foi citado pelo entrevistado também que os parâmetros definidos para consulta são muitos, podendo, em sua opinião, serem mais simples, entrando apenas com a data desejada e uma faixa de horário para reduzir o número de registros buscados.

As classes presentes nesse trabalho foram reutilizadas do TimCam (SOURCEFORGE, 2003). Para a parte do envio de imagens via FTP não foram alteradas as classes responsáveis, que ficam no pacote `ftp`, devido ao fato que o tratamento necessário como, a conexão, a autenticação e envio, já estavam plenamente atendidas.

Durante a fase de desenvolvimento, foi verificado que são três os itens relevantes a serem controlados, para ser possível alcançar o processo de comparação de imagens de forma satisfatória, sendo eles: os equipamentos utilizados, o nível de tolerância e a iluminação do ambiente.

Os equipamentos utilizados, especialmente a câmera, são fatores críticos no desempenho do sistema no processamento das imagens, ele foi feito para trabalhar com poucos recursos, mas é necessário que consiga processar as imagens em tempo hábil, conforme os parâmetros configurados. Para os testes foi utilizado um notebook com processador Intel Core2Duo de 1.66GHz e 2gb de memória RAM e uma *webcam* KMEX modelo AW-A1030, que tem resolução máxima de 640x480 pixels, formato de captura de imagem *Joint Photographic Experts Group* (JPEG) ou *BitMap* (BMP), formato da imagem

RGB24, com 24 *bits* por pixel (KMEX, 2010).

O tempo de processamento da comparação das imagens foi obtido usando a função `System.currentTimeMillis()` que é nativa do Java, e nos seus resultados pôde-se observar que o tempo não ultrapassou os 140 milisegundos, tornando o projeto viável para o objetivo proposto. O tempo máximo do processamento obtido foi sempre menor que o tempo mínimo configurável entre as capturas, que é de um segundo. Os recursos de hardware citados, que foram utilizados para os testes, suportaram o processamento sem que fosse registrado nenhum travamento do protótipo, do computador e sem atraso entre a comparação das imagens.

O tamanho de cada imagem gravada varia de 5 a 9 KBytes (KB). Para dar uma melhor noção do que isso reflete, utilizando um disco rígido com espaço disponível de 300 gigabytes, e considerando que é gravada uma imagem de 5kb em intervalos de 5 segundos, tem-se 62.914.560 imagens que podem ser armazenadas. Isso no período de 3.641 dias, ou 9 anos e 11 meses, ou seja, ainda desconsiderando o recurso da comparação e eliminação o protótipo já está com uma economia de recurso de disco, considerando o período citado acima, como um longo tempo em termos de gravação de dados e também o pequeno tamanho que as imagens possuem, conforme obtido nesse exemplo. Para ser usado em situações reais, a captura ideal seria a cada 1 segundo, para evitar que algo passe despercebido.

O segundo componente crítico de protótipo, denominado de nível de tolerância, é o ponto principal do algoritmo utilizado na comparação de imagens. O nível de tolerância, conforme a própria denominação, determinará qual será o nível de variação tolerado na comparação das imagens, para que elas sejam consideradas semelhantes ou diferentes, afetando diretamente a necessidade de mais ou menos espaço em disco para a gravação das imagens.

O nível de tolerância é formado pelo resultado válido, ou não, na comparação entre duas imagens, para isso são feitas duas validações. A primeira validação verifica se um determinado pixel na primeira imagem é semelhante ao mesmo pixel na segunda imagem, sendo considerado semelhante quando na escala RGB, os dois pixels variarem dentro do percentual configurado pelo usuário no protótipo.

A segunda validação é feita através da comparação do número de pixels válidos de uma imagem para outra, essa validação é positiva quando o número total de pixels válidos na primeira comparação for superior a noventa por cento. Esse valor de noventa por cento é pré-fixado e foi identificado, nos testes, como sendo aquele que retornava os melhores resultados na comparação das imagens, abaixo disso a diferença já é visível e pode ser considerada relativa, dependendo do que está sendo monitorado. O Quadro 12 apresenta o pseudocódigo

com a fórmula para a primeira validação, sendo que `m` recebe o menor valor RGB dos dois pixels comparados e `M` recebe o maior valor RGB dos dois pixels. O `m` é então dividido pelo `M` e multiplicado por cem, se esse percentual for maior ou igual ao percentual de nível de tolerância definido pelo usuário o totalizador `Pontos` é incrementado em um.

```
Se {[(m /M) * 100] >= Nível_de_Tolerância_configurando_pelo_usuário}então
Inicio
    Pontos ++;
Fim.
```

Quadro 12 – Pseudocódigo com fórmula para o cálculo do nível de tolerância

Em seguida (Quadro 13) é feita a segunda validação onde é comparado o total de pixels válidos, totalizados no contador `Pontos` com o total de pixels da imagem, se o total de pixels válidos for igual ou maior que noventa por cento, a imagem será considerada semelhante e não será gravada.

```
Se (Pontos / Total_de_pixels) >= 90% Então
  Inicio
   // as imagens são semelhantes, apaga e passa para a próxima comparação.
  Senão
   // as imagens não são semelhantes, faz gravação e envio da imagem;
Fim
```

Quadro 13 – Pseudocódigo com fórmula que define semelhança

A iluminação do ambiente foi o fator que mais gerou implicações na execução dos testes e validação, pois se verificou nos primeiros testes que o tipo da luz e as ondas emitidas por elas eram captadas pela câmera, e alteravam as proporções do valor do RGB do pixel.

No Quadro 14 abaixo, encontra-se a comparação de valores percentuais obtidos em um mesmo ambiente, num teste comparando 10 imagens para cada tipo de iluminação, numa coluna estão os valores percentuais com iluminação fluorescente e na outra com iluminação de luz natural do dia, o percentual de tolerância definido para os testes foi 95%.

Quadro de Comparação entre Tipos de Iluminação		
Imagens comparadas	Fluorescente	Luz Natural
	% de semelhança	% de semelhança
Imagem 1	-	-
Imagem 2	91,16	99,35
Imagem 3	94,83	99,57
Imagem 4	93,01	99,68
Imagem 5	92,82	98,12
Imagem 6	79,80	99,20
Imagem 7	91,81	98,91
Imagem 8	95,54	99,28
Imagem 9	94,92	98,53
Imagem 10	87,17	99,74

Quadro 14 – Variação do percentual de semelhança em ambientes com luz natural e fluorescente

O ambiente iluminado com lâmpadas fluorescentes foi o que mais gerou resultados variáveis, devido principalmente às ondulações que eram captadas pela *webcam*, resultando em percentuais abaixo dos 90% da margem de semelhança, com maior freqüência.

Como exemplo disto, na Figura 22 e Figura 23 é visível que ondas emitidas pela luz fluorescente são captadas pela câmera. Consequentemente, ao serem comparadas uma com outra, tiveram seus valores de RGB alterados, ficando fora da margem de tolerância com um valor menor que 90% de semelhança entre as imagens. Com isso a imagem_2 exibida na Figura 23 foi gravada em banco, mesmo sem ter alterações relevantes, em comparação com a figura anterior.

Por esse motivo o percentual de tolerância pode ser alterado dinamicamente, pois num ambiente como o testado no exemplo, o valor pode ser diminuído e com isso as pequenas oscilações captadas podem ser ignoradas, assim, apenas as diferenças que forem consideradas relevantes, como a movimentação de pessoas, por exemplo, será captada e devidamente registrada pelo protótipo.

Figura 22 – Imagem_1 com ondulações captadas

Figura 23 – Imagem_2 com ondulações captadas

Pôde-se verificar na Figura 22e 23 ondas que são captadas e são perceptíveis a olho nu, mesmo o ambiente não sofrendo nenhuma alteração por elementos sendo capturados pela câmera. Tais elementos seriam pessoas passando, luzes sendo apagadas ou acesas, ou outros.

A oscilação das ondas influencia negativamente no funcionamento, por alterar os valores de RGB e gravando imagens que não são consideradas relevantemente diferentes.

O primeiro teste efetuado em ambiente com luz natural, ou seja, com luz do dia, os resultados obtidos ficavam mais próximos de 100% de semelhança, como no exemplo da Figura 24, que o percentual da semelhança variava na faixa de 99% entre uma imagem e outra. Com a luz de lâmpadas incandescentes, o resultado das comparações ficou satisfatório, com uma porcentagem mais baixa que a luz natural, mas também oscilando na margem de 90 a 100% de semelhança.

No teste da Figura 24 encontram-se quatro imagens tiradas em uma sala com iluminação natural, ou seja, apenas a luz do sol, tiradas num intervalo de 5 segundos, chamadas de `imagem_1, imagem_2, imagem_3 e imagem_4`. Foi possível verificar que apesar de parecerem iguais, elas não são e possuem pequenas diferenças de tonalidades, geradas pela luminosidade do ambiente.

Sem o nível de tolerância todas elas seriam gravadas, mas devido aos cálculos e a margem de comparação, a `imagem_1` foi gravada, por padrão do protótipo, em seguida, da primeira imagem para a segunda, verificou-se que 99,35% dos pixels se mantiveram dentro da margem de comparação, com isso não foi gravada, da segunda para a terceira 99,53% de semelhança e da terceira para a quarta 99,50 % de semelhança.

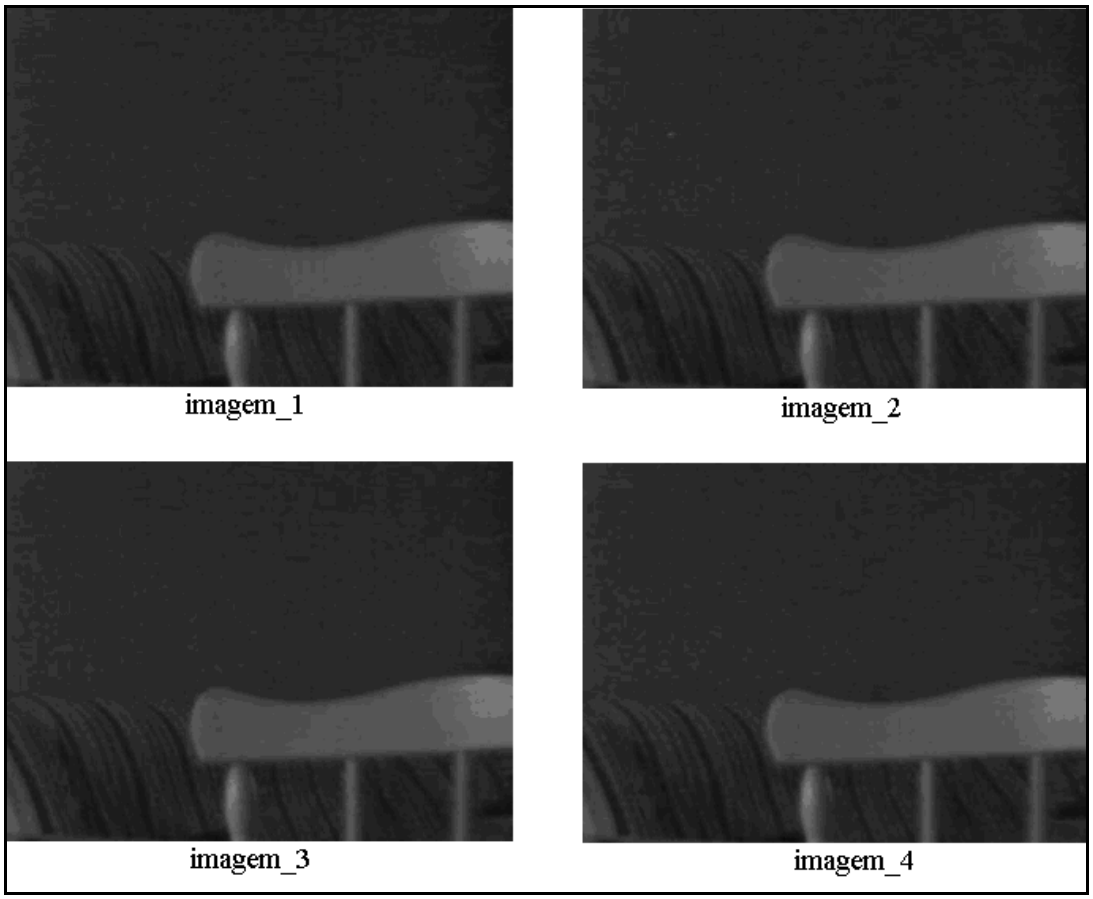


Figura 24 – Comparação entre imagens em ambiente com luz do dia

No gráfico da Figura 25 pode-se ter uma melhor visualização dos percentuais obtidos na comparação em ambiente de iluminação natural do dia.

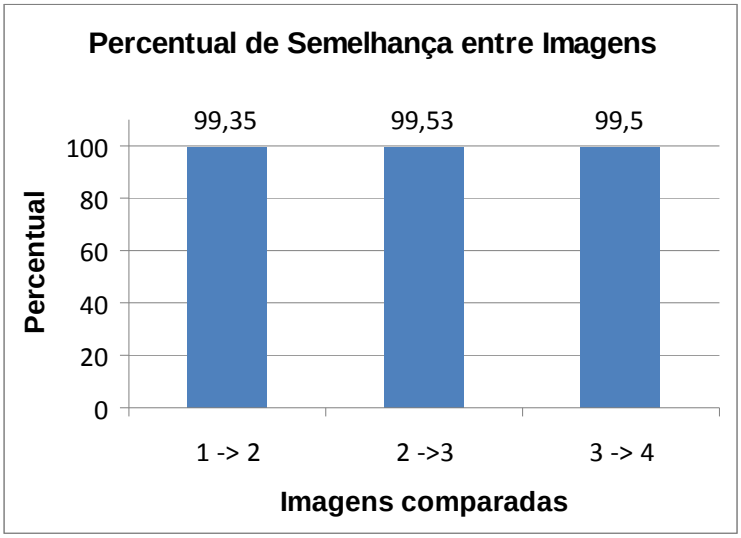


Figura 25 – Gráfico com percentual de comparação por imagens em ambiente com luz natural

Agora é apresentado um novo teste conforme Figura 26, mas com uma variação proposital no ambiente monitorado, com a inclusão de um elemento na frente da câmera, para que o valor do RGB sofresse alteração e a comparação ficasse fora da margem de semelhança, isso, para poder exemplificar a funcionalidade do protótipo. O ambiente de teste agora, era

iluminado por lâmpadas fluorescentes.

As imagens da Figura 26 comparadas agora, são chamadas de `imagem_5`, `imagem_6`, `imagem_7` e `imagem_8`, e o teste se deu da seguinte forma, o protótipo foi iniciado, e as três primeiras imagens foram tiradas sem nenhuma alteração no ambiente e a quarta com uma alteração proposital. Dessa forma é dada a noção do que é considerado fora da margem da comparação.

Na `imagem_8`, o ambiente foi alterado com a adição de uma mão que aparece no canto da imagem, reduzindo consideravelmente o percentual de semelhança, tendo como resultado da comparação da `imagem_5` para a `imagem_6` o total de 92,72% dentro da margem tolerada, da `imagem_6` para a `imagem_7`, 92,59% de semelhança, da `imagem_7` para a `imagem_8` já baixou para 62,36% o percentual de semelhança, e a imagem foi gravada.

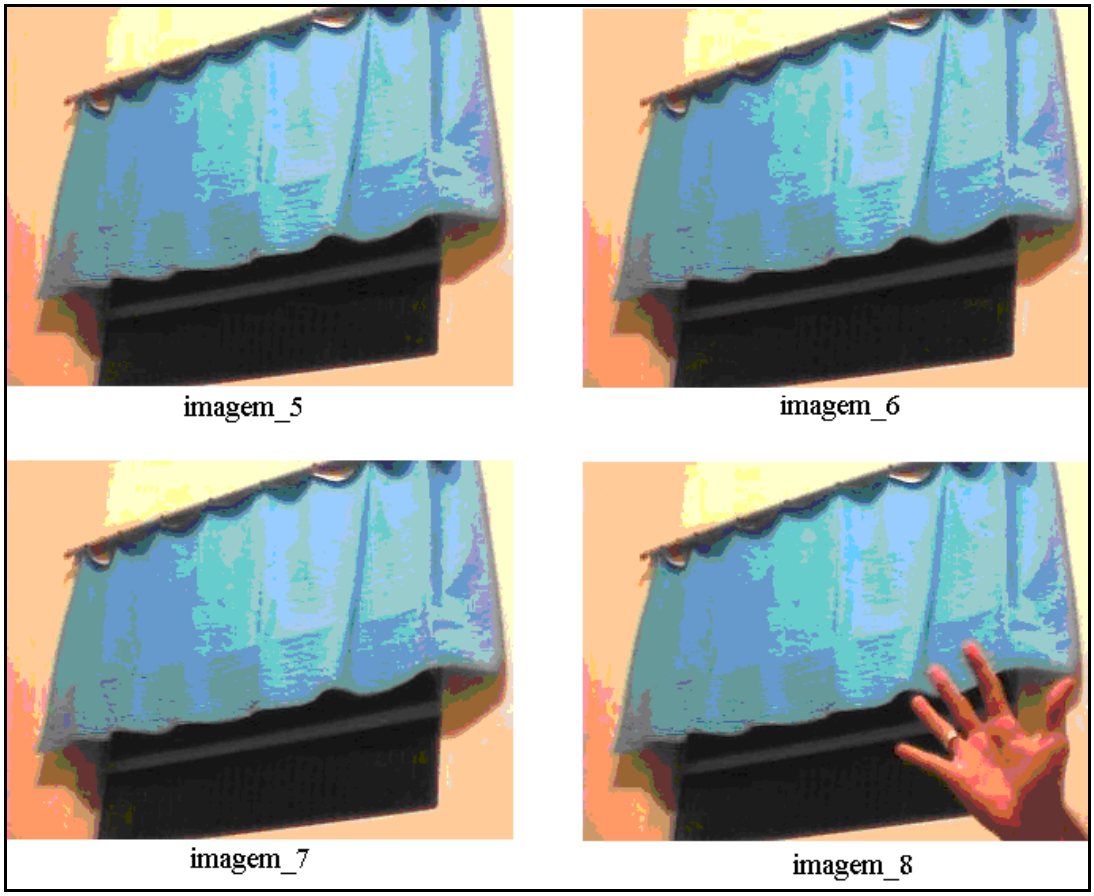


Figura 26 – Comparação entre imagens com iluminação fluorescente

No gráfico da Figura 27 pode-se ter uma melhor visualização da comparação das imagens contidas na Figura 26, através dos valores percentuais gerados. Pode-se notar pelos valores percentuais de semelhança entre as imagens, que são mais baixos comparados ao teste da Figura 24, esse fato ocorre principalmente em ambientes com lâmpadas fluorescentes.

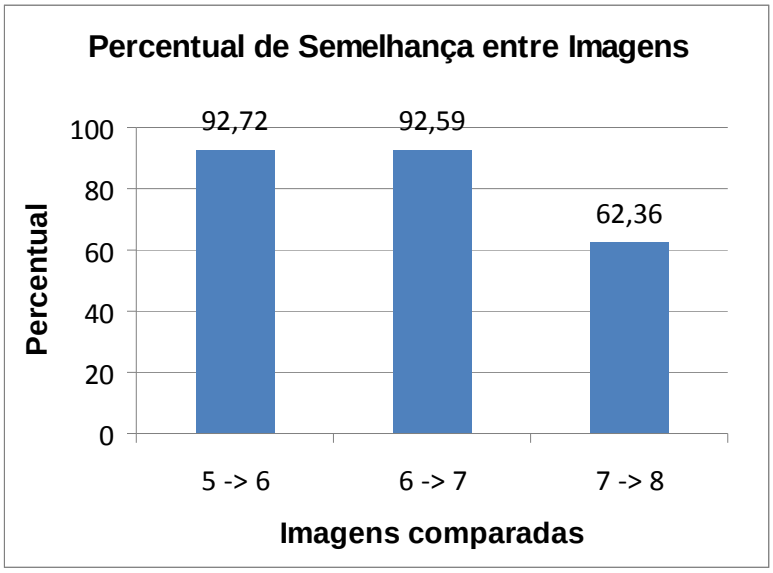


Figura 27 - Gráfico com percentual de comparação por imagens em ambiente com luz fluorescente

A Figura 26 com as imagens comparadas foi propositalmente deixada com qualidade baixa, a mesma foi copiada no MS-Paint Brush e salva como bitmap de 256 cores, para que possa ser visualizado que não são as mesmas.

Quando vistas na resolução normal que são tiradas, que é a padrão da *webcam*, citada no começo da seção 3.4, esse detalhe não fica perceptível ao olho nu.

Como último teste descrito, é executada no protótipo uma seqüência de captura e comparação entre 20 imagens, captadas no intervalo de cinco segundos. O ambiente de testes foi uma sala com iluminação com lâmpadas incandescentes, pois assim como o ambiente com luz natural do dia, se mostrou mais preciso para o resultado das comparações, anulando o problema com as ondulações.

No Quadro 15 abaixo são demonstrados os detalhes do teste, com dados da imagem, percentual de semelhança, as imagem que foram gravadas ou excluídas, com destaque para as únicas imagens gravadas em banco de dados.

O percentual de tolerância para o valor de RGB definido para este teste foi de 95% e o tamanho de cada imagem ficou em 7KB. A primeira imagem foi gravada por padrão, depois disso as imagens 6 e 10 foram gravadas, pelo fato que na sala onde o teste foi executado no momento da captura, houve uma oscilação do vento que movimentou a cortina.

Com isso o valor RGB sofreu uma alteração considerável, ficando abaixo dos 90 por cento de semelhança, quando comparada com a imagem anterior. Isso demonstra uma eficácia deste processo no que diz respeito ao recurso de economia de espaço em disco.

Em seguida ao Quadro 15 encontra-se o detalhamento do teste com os percentuais de obtidos em relação a espaço x economia.

Imagem	Percentual de semelhança	Imagem gravada
Imagem 1	-	**Sim**
Imagem 2	91,16	Excluída
Imagem 3	94,83	Excluída
Imagem 4	93,01	Excluída
Imagem 5	92,82	Excluída
Imagem 6	79,80	**Sim**
Imagem 7	91,81	Excluída
Imagem 8	95,54	Excluída
Imagem 9	94,92	Excluída
Imagem 10	87,17	**Sim**
Imagem 11	94,95	Excluída
Imagem 12	95,17	Excluída
Imagem 13	94,79	Excluída
Imagem 14	94,11	Excluída
Imagem 15	94,71	Excluída
Imagem 16	94,72	Excluída
Imagem 17	94,56	Excluída
Imagem 18	94,91	Excluída
Imagem 19	94,83	Excluída
Imagem 20	94,65	Excluída

Quadro 15 – Resultado da comparação e exclusão de imagens durante 00:01:20h

Neste teste pode-se verificar principalmente o que gerou de economia, por ter eliminado 17 do total de 20 imagens capturadas, considerando que as imagens obtidas ficaram com tamanho de 7kb cada. No período de um minuto e vinte segundos, a economia conseguida foi de 119kb de espaço em disco, ou seja, de 85%, em uma hora o valor desta economia já se mostra considerável, considerando essa proporção conseguida em um minuto e vintes segundos, a economia seria de 5.355 megabytes. Ainda se fosse considerado que o protótipo tivesse mais de uma câmera, o espaço consumido estimado seria multiplicado pelo número total de câmeras contidas. Neste caso, mesmo com a eliminação de imagens, tendo mais câmeras o espaço disponível para armazenamento já precisaria ser maior, para garantir um bom tempo de armazenagem, tal qual o exemplo citado no começo desta seção.

Em suma, para ambientes externos devido ao grande número de elementos que podem estar envolvidos, seja o vento que modifica posições de objetos ou crie sombras e reflexos, animais domésticos cruzando em frente à câmera, oscilações constantes de luz, entre outros, a utilização do RGB como modelo de cor, não permitiu refinar a comparação das imagens.

Com a possibilidade de alterar o percentual de tolerância dinamicamente através da interface, o protótipo pode ser adaptado para o ambiente que estiver sendo monitorado, sendo o valor aumentado caso a iluminação seja natural ou incandescente, ou diminuído caso

fluorescente. Considerou-se os ambientes fechados, com base nos testes, como o ideal para a utilização do protótipo.

4 CONCLUSÕES

O presente trabalho possibilitou desenvolver um protótipo de monitoração utilizando uma *webcam*, capaz de fazer uma monitoração satisfatória de um ambiente fechado, considerando que o ambiente seja o ideal, definido nos testes da seção anterior.

O protótipo possui na interface os recursos mais utilizados num sistema de CFTV, que entre os estudados para o desenvolvimento, são a possibilidade de uma consulta aos registros com a visualização das imagens na própria tela do protótipo. Também a possibilidade de acesso remoto às imagens gravadas via um servidor com acesso FTP e por fim, baixo custo por utilizar poucos recursos hardware e software.

Como recurso de hardware, pode-se citar o tamanho do disco rígido, que pelo fato de as imagens possuírem pequeno tamanho, não requer que este seja além do normal encontrado em computadores domésticos, que variam normalmente de 200 a 360 gigabytes. A configuração do computador também não requer grande capacidade de processamento, como já demonstrado na seção 3.4.

Como requisito do protótipo, foi possível desenvolver uma comparação de imagens, onde um algoritmo analisa a similaridade das imagens capturadas. Se as duas imagens que são comparadas estiverem dentro do nível de tolerância, e uma delas em relação a outra estiver acima da margem de semelhança, definida como ideal nos testes, então uma delas é eliminada.

O protótipo mostrou-se eficaz no que diz respeito à comparação de imagens, quando testado em condições de luz favorável, ou seja, de uma luz não fluorescente de preferência e em ambientes fechados, como, salas comerciais, escritórios, interior de residências, entre outros.

Em ambientes externos ocorrem constantes alterações por variações com luz do sol e sombras geradas, ou por condições climáticas, como vento, que podem causar grandes mudanças no conteúdo da imagem, sem que realmente seja um acontecimento que necessite que mais uma imagem seja gravada. Outro fator alcançado no desenvolvimento do protótipo foi que, com a possibilidade de eliminar as imagens semelhantes, os recursos de hardware são economizados, tratando-se do espaço em disco.

O trabalho, porém, possui suas limitações, por exemplo, quanto maior a área de cobertura da câmera, maior a chance de uma comparação abaixo do nível de tolerância, pelo fato de ter uma maior chance de alterações nos tons da imagem e que o algoritmo acabe por

considerá-las diferentes, mesmo que não tenham alterações visíveis ao olho nu.

O padrão RGB também é mais suscetível a variações e consequentemente mais difícil de definir padrões de igualdade entre imagens e o ambiente influencia diretamente no resultado, um outro padrão poderia alcançar resultados mais eficazes.

Um fator que torna a necessidade de um melhor tratamento para o código de comparação de imagens, é o fato que, a cada dia é mais comum o uso de lâmpadas fluorescentes devido a sua economia de energia. Esse importante detalhe reforça mais ainda a necessidade de modelo de cor, que permita obter um novo parâmetro para a margem de semelhança, considerando principalmente a ondulação captada pela câmera, oriunda da iluminação fluorescente, como ocorreu nos testes obtidos no protótipo. O modelo de cor que foi observado como melhor substituto ao RGB é o YCbCr, citado na seção 2.5.1.

4.1 EXTENSÕES

O protótipo apresentado permite que sejam realizadas novas alterações e adição de novas funcionalidades, sendo uma delas a utilização de outro padrão de cores para a comparação de imagens ao invés do RGB, acreditando dessa forma, chegar mais próximo da taxa de 100% de acerto na comparação. Podendo esse ser o YCbCr, que teria mais variantes de cor para se trabalhar, utilizando parâmetros de intensidade e não apenas um valor para a cor do pixel.

Considerando que, uma das dificuldades encontradas foi definir uma margem de semelhança ideal das imagens, principalmente pelo motivo da iluminação do local. Desta forma, novos testes podem ser feitos a fim de encontrar uma melhor maneira de tratar o fator luminosidade.

Também pode ser verificado se a partir do que já foi obtido neste protótipo, existe a possibilidade de monitorar mais de um ambiente, com a adição de duas ou mais *webcams*, para isso teria que fazer um estudo mais detalhado da estrutura do JMF, no que diz respeito à forma de captura de um dispositivo de câmera. Caso seja implementado a captura por mais de uma *webcam*, seria necessário também um tratamento para a nomenclatura das imagens, considerando que teoricamente cada câmera monitoraria um ambiente.

As imagens poderiam ser salvas com nomes que possibilitem identificar de onde foram capturadas, uma idéia seria a inclusão da identificação do ambiente na montagem do nome da

imagem, como por exemplo, sala-dia/mês/ano, escritório-dia/mês/ano e assim por diante.

Alterações de configuração da imagem da câmera também poderiam ser adicionadas, caso o usuário queira aumentar ou diminuir brilho, contraste, dando assim uma possibilidade ao usuário de definir com qual qualidade ele quer que imagens sejam capturadas e gravadas.

Outra opção seria a adição de mais portas de comunicação para a parte do envio via FTP, para o tratamento de falhas, caso uma esteja indisponível, teria outra opção a ser usada e o envio não seria interrompido.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBUQUERQUE, Rafael B. **Esteganografia: Análise de algoritmos baseada em comparação entre imagens**. [S.l.], 2008. Disponível em: < http://dsc.upe.br/~tcc/20082/TCC_Rafael-Esteganografia_vf.pdf>. Acesso em: 16 out. 2010.

DIGITAL SECURY. **Vantagens do cftv?.** Rio de Janeiro, 2008. Disponível em: <http://www.digitalsecury.com.br/vantagens.html>. Acesso em: 18 ago. 2010.

EDITORA ABRIL. **Socorro, a criminalidade apavora a classe média**. São Paulo, 2000. Disponível em: <http://veja.abril.com.br/070600/p_132.html>. Acesso em: 19 ago. 2010.

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento de imagens digitais.** Tradução de Roberto Marcondes Cesar Junior, Luciano da Fontoura Costa. São Paulo: Edgard Blücher, 2000.

GOUVEIA, Daniel. **Comunicações Multimídia na internet: Da teoria a pratica.** Rio de Janeiro: Editora Ciência Moderna Ltda., 2008.

GUIA DO CFTV. **Sistema de cftv digital**. Porto Alegre, 2008. Disponível em: <http://www.guiadocftv.com.br/modules/smartsection/item.php?itemid=17> Acesso em : 18 ago. 2010.

JAVAWORLD. **Program multimedia with JMF**. Framingham, 2009. Disponível em: < http://www.javaworld.com/jw-04-2001/jw-0406-jmf1.html?page=1>. Acesso em: 18 ago. 2010.

KMEX. **Acessórios/WebCam.** [S.l.], 2010. Disponível em: < http://www.k-mex.com.br/WebForms/ProductInfoPage.aspx?KindNo=0000000010&ProductNo=0000000082>. Acesso em: 30 out. 2010.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **Processamento Digital de Imagens**, Rio de Janeiro: Brasport, 1999.

PAULA FILHO, Wilson de Pádua. **Multimídia conceitos e aplicações.** Rio de Janeiro: LTC, 2000.

Sindicato das empresas de segurança privada – Rio de Janeiro. **Convenção coletiva de trabalho 2009/2010.** Rio de janeiro, 2009. Disponível em: < http://www.sindesp-rj.com.br/frame/convencoes/convencao_coletiva_de_trabalho.pdf>. Acesso em: 30 set. 2010.

SOURCEFORGE. **TimCam**. [S.1], 2003. Disponível em: < http://timcam.sourceforge.net/>. Acesso em: 22 out. 2010.

SOUZA, Jean Carlos de. **Funções essenciais de um CFTV**. Blumenau, 12 nov. 2010. Entrevista concedida a Ederson José na empresa Cecred.

SUN MICROSYSTEMS. **JMF Guide**. 2008. Disponível em: < http://www.oracle.com/technetwork/java/javase/tech/index-jsp-140239.html >. Acesso em: 10 ago. 2010.

TELEALARME. **Gvi – Sistema CFTV**. Blumenau, 2008. Disponível em: <http://www.telealarme.com>. Acesso em: 11 ago. 2010.

VEJA SUA SEGURANÇA. **O melhor cão de guarda**. São Paulo, 2001. Disponível em: <http://veja.abril.com.br/especiais/seguranca/p_086.html>. Acesso em: 19 ago. 2010.

MULTIMEDIAWIKI. **YCbCr**. 2009. Disponível em <http://wiki.multimedia.cx/index.php?title=YCbCr> . Acesso em: 22 nov. 2010.

YOICS. **Access and manage your computers and network devices from anywhere.**. Califórnia, 2008. Disponível em: < http://www.yoics.com/>. Acesso em: 19 ago. 2010.