UNIVERSIDADE FEDERAL DE ALFENAS
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Gabriel Lara De Carli

SOFTWARE DE SEGURANÇA DE BAIXO CUSTO UTILIZANDO UMA *WEBCAM*

Alfenas, 13 de novembro de 2015.

UNIVERSIDADE FEDERAL DE ALFENAS
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

SOFTWARE DE SEGURANÇA DE BAIXO CUSTO UTILIZANDO UMA *WEBCAM*

Gabriel Lara De Carli

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Alfenas, como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Orientador: Prof. Luiz Eduardo da Silva.

Alfenas, 13 de novembro de 2015.

Gabriel Lara De Carli

SOFTWARE DE SEGURANÇA DE BAIXO CUSTO UTILIZANDO UMA *WEBCAM*

A Banca examinadora abaixo-assinada aprova a monografia apresentada como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de Alfenas.

Prof. Flávio Barbieri Gonzaga

Universidade Federal de Alfenas

Prof. Vinícius Ferreira da Silva

Universidade Federal de Alfenas

Prof. Luiz Eduardo da Silva (Orientador)

Universidade Federal de Alfenas

Alfenas, 13 de novembro de 2015.

.

.

AGRADECIMENTO

Eu agradeço à minha família pelo suporte do início ao fim da graduação. Agradeço à minha namorada Laís pela ajuda e compreensão. Agradeço também aos meus melhores amigos Rodrigo Kairala, Renan Barros e Thadeu Carvalho por todo auxílio, contribuindo de forma direta ou indireta para minha formação.

"No que diz respeito ao desempenho, ao compromisso, ao esforço, à dedicação, não existe meio termo. Ou você faz uma coisa bem-feita ou não faz. "

(Ayrton Senna)

RESUMO

Com o aumento da criminalidade, a população tem buscado formas de proteger seus bens pessoais, e um desses caminhos é a instalação de sistema de vigilância que monitore o ambiente desejado. Porém, tais sistemas tendem a ter um alto custo, tanto para aquisição dos recursos quanto para a implementação do *software*. Este trabalho apresenta um sistema de segurança que utiliza como ferramenta de monitoramento uma *webcam*. A intenção é apresentar um sistema multiplataforma de baixo custo para que a população em geral possa ter acesso. O desenvolvimento do projeto teve como base a linguagem de programação *Java*, utilizando a biblioteca *OpenIMAJ*. Para economia de recurso, foi implementada uma classe que permite a gravação do vídeo a partir do momento em que a lente da câmera capta um movimento e é interrompida no momento em que o ambiente se torna estático, fazendo com que ocupe o menor espaço possível em disco. Ao ser reconhecida uma pessoa, o sistema salva a imagem do rosto com o intuito de permitir a identificação

Palavras-Chave: segurança, *Webcam*, detecção de faces, *openIMAJ*.

ABSTRACT

With the criminal increase, people have looking for ways to protect your personal goods, and one of those ways is the surveillance system that monitors the desired environment. However, these tend to be expensive, both to purchase the resources and to implement the software. This work presents a security system that uses a webcam as monitoring tool. The intention is to present a low-cost platform system so that the general public may have access. The project development was based on the Java programming language, using OpenIMAJ library. For resource saving, a class that allows the video recording was implemented from the time when the lens picks up a move and stop at the time when the environment becomes static, to occupy the smallest possible space on disk. When a person is found, the system saves the face image in order to allow identification.

Keywords: security, Webcam, face detection, openIMAJ.

LISTA DE FIGURAS

LISTA DE ABREVIAÇÕES

FPS	*Frames Per Second*
HD	*Hard Disk*
IBGE	Instituto Brasileiro de Geografia e Estatística
IDE	*Integrated Development Environment*
JVM	*Java Virtual Machine*
JRE	*Java Runtime Environment*
POM	*Project ObjectModel*
SMS	*Short Message Service*
USB	*Universal Serial Bus*
XML	*Extensible Markup Language*
DVR	*Digital Video Recorder*
GSM	*Global System for Mobile Comunications*
IP	*Internet Protocol*

SUMÁRIO

1
Introdução

Nos últimos anos, as estatísticas criminais têm mostrado um aumento considerável no Brasil. Roubos seguidos de morte (latrocínios) aumentaram 28,6% de 2010 a 2013, segundo o Relatório Consolidado de Ocorrências de latrocínios registrados pela Polícia Civil (BRASIL. MINISTÉRIO DA JUSTIÇA, 2013). Num período de 15 anos (2000-2014), o estado de São Paulo, o mais populoso do Brasil, registrou aumento no número de roubos: de 581 para 726/100 mil habitantes (BRASIL. GOVERNO DO ESTADO DE SÃO PAULO, 2015). Dessa forma, com o aumento da violência, a população tem buscado por sistemas de segurança particular (MIGUEL, 2007).

Atualmente, existem vários sistemas de segurança disponíveis no mercado, como o *Sighthound*. Esse sistema é eficaz no reconhecimento de indivíduos e objetos, como veículos (SIGHTHOUND, 2015). Entretanto, apresenta um alto custo e sua venda é feita online sem conteúdo em português, podendo representar um obstáculo para a população de baixa renda. Assim, apenas a parcela que possui uma maior renda consegue adquirir sistemas de segurança de qualidade para a moradia ou estabelecimento comercial (MIGUEL, 2007).

Outros sistemas de segurança amplamente utilizados pela população são os alarmes e cercas elétricas. Porém, esses sistemas não registram imagens para a obtenção de provas concretas e identificação do infrator. Sistemas de vigilância com registro de imagens são fundamentais para que o Estado aplique as leis penais cabíveis, contribuindo, assim para a diminuição das ocorrências criminais.

Atualmente, o desenvolvimento de sistemas de segurança utilizando equipamentos disponíveis em domicílios e/ou de fácil acesso está em ascensão, e a detecção de movimento por meio de sinais de vídeo têm forte apelo de uso nesses sistemas (WANG et al., 2004).

Este trabalho tem como finalidade propor um sistema de vigilância de baixo custo, que seja viável financeiramente para que a população em geral possa adquirir, tendo como principal método o armazenamento de imagens relevantes à detecção de movimentos.

1.1 Justificativa e Motivação

A crescente demanda por sistemas de vigilância tem estimulado o surgimento de novos equipamentos e pesquisas por *softwares* que apresentam melhor qualidade e desempenho. Porém, o produto final acaba sendo repassado ao cliente com custo elevado, limitando o acesso a esse tipo de sistema apenas às pessoas com situação financeira elevada.

Apesar da alta demanda por mecanismos de análise automática de sequências de vídeo para detecção de movimento, há uma escassez de soluções de baixo custo, tendo como consequência a inacessibilidade a esses mecanismos pela maior parte da população (MATTOS; ANDREATTA, 2010).

Outro problema é a falha nos algoritmos para armazenamento de imagens com relevância, que acabam ocupando espaço no HD sem necessidade, obrigando o usuário final adquirir HD com maior capacidade de armazenamento.

Existe ainda a necessidade de reconhecimento facial do indivíduo a ser detectado pelo sistema, para a apreensão do suspeito, caso ocorra um crime.

Este projeto tem o intuito de oferecer segurança acessível à maior parte da população, minimizando os custos de equipamentos de monitoramento. Como atualmente cerca de 49,5% dos brasileiros possui computadores em seus domicílios (dados divulgados pelo IBGE no dia 18/09/2014), sistemas de monitoramento que necessitem apenas de uma câmera de baixa resolução (*webcam*), e que apresentem o mínimo de segurança contra um intruso ou um ladrão, pode ter uma boa receptividade pela a população em geral isto é, desde um simples estudante

universitário, que deseja proteger sua moradia durante sua ausência, até um microempresário, que se preocupa com seu pequeno estabelecimento comercial, poderão adquirir este sistema.

1.2 Problematização

Diante da justificativa apresentada, a questão que deverá ser respondida no desenvolvimento desse trabalho é: será possível desenvolver um sistema de segurança eletrônico usando recursos tecnológicos baratos como uma *webcam*?

1.3 Objetivos

1.3.1 Gerais

Desenvolver um sistema de segurança eletrônica usando recursos tecnológicos acessíveis à população de baixa renda.

1.3.2 Específicos

- Desenvolver um sistema com identificação de movimentos para capturar sequências de imagens por meio de uma *webcam*;
- Implementar algoritmos para extrair das imagens capturadas faces de possíveis suspeitos;
- Comprimir as imagens das faces e o envio dos arquivos para o e-mail do utilizador desse sistema;

- Analisar e discutir os resultados obtidos com o software que será desenvolvido.

1.4 Organização da Monografia

No Capítulo 2, estão os métodos básicos para o sistema de segurança e os produtos existentes no mercado. O Capítulo 3 apresenta a metodologia utilizada para o desenvolvimento do projeto. O Capítulo 4 descreve como o projeto foi desenvolvido. Os resultados do trabalho são encontrados no Capítulo 5.

2
Sistemas de Segurança

2.1 Considerações Iniciais

Este trabalho pode ser considerado um pequeno auxílio perante alguns problemas vistos no dia-a-dia juntamente com muitas soluções existentes no mercado sendo, a maioria delas, de alto custo e possivelmente não acessível a grande parcela da população brasileira.

Tendo em vista a praticidade e baixo custo de uma *Webcam*, é possível a criação de um eficiente módulo de detecção de movimentos e localização de faces, aliados a gravações de vídeos. A gravação ocorrerá apenas a partir da detecção de movimentos, ocorrendo assim, a economia de espaço da memória secundária e difere de muitas soluções existentes no mercado, que gravam sem parar e sem qualquer tratamento de imagem e vídeo.

Em relação aos sistemas de mercado, muitos comércios e lojas utilizam DVR que consiste basicamente em um conjunto de câmeras e uma central física composta por um ou mais discos rígidos responsáveis por armazenar todo o vídeo gerado pelas filmagens. Uma grande desvantagem é a central com a gravação ficar exposta, podendo ser furtada por algum indivíduo que entra indevidamente no local. Soluções "na nuvem" para esse tipo de problema acabam se tornando fundamental. E também, existem os que se baseiam em soluções automatizadas de grande porte, porém há certa resistência em relação à confiabilidade perante o público, visto que são complexos e existe o receio de não funcionar. Outro fator decisivo neste quesito é

o fato da residência ou comércio não suportar tal equipamento, necessitando de uma reforma física no local a ser implantado (MEDEIROS, 2009).

Já outros, se baseiam em monitoramentos visuais envolvendo grande conhecimento em redes e eletrônica, no que tange criação de conta em site como DynDNS[1], sensores infravermelhos e circuito de controle de motor (NERES, 2007).

E também há os sistemas clássicos na área da segurança, como os alarmes. Mas ainda estes, apesar da grande variedade de preço no mercado, o custo continua sendo um fator relevante na escolha (GUADALUPE, 2014).

É vísível, num modo geral, nos sistemas de segurança avaliados, a falta de desenvolvimento de algoritmos para melhorar o uso desses recursos tecnológicos a fim de obter sistemas que usem poucos recursos e sejam de baixo curso.

2.2 Revisão Bibliográfica

O trabalho de Katie Hafner (2003), apresenta uma proposta de um sistema de segurança em escolas usando *WebCam*. Nesse trabalho, as *Webcams* estão instaladas nos corredores de uma escola e todas as pessoas envolvidas diretores, alunos, pais de alunos, podem acessar as imagens obtidas de maneira remota através da posse de um *password*. Segundo a autora, esse trabalho teve uma boa aceitação da comunidade, incluindo administradores, professores e estudantes. No entanto, nenhum tratamento é utilizado nas imagens obtidas, são simplesmente disponibilizadas sem o uso de qualquer algoritmo como, por exemplo, um algoritmo de deteção de faces.

No artigo de Lin, Lee (2012), o autor desenvolve um projeto para mapeamento da íris do olho humano. Também discute se é possível criar um algoritmo de alta

[1] DynDNS é um exemplo de DNS (sistema de nomes de domínio), que é um banco de dados distribuído implementado em uma hierarquia de servidores de nome (KUROSE; ROSS, 2011).

performance e baixo custo computacional e obter êxito utilizando imagens de *Webcam*, desde que a iluminação esteja boa. Neste trabalho, Lin e Lee também utilizam o recurso da Imagem Integral[2] para que a localização de uma face (a fim de localizar os olhos) seja calculada de modo rápido. Porém, os cálculos posteriores são focados nos reconhecimentos e moldagem dos olhos humanos. Apesar do processamento primário ser semelhante ao proposto neste projeto, difere nos objetivos finais, no qual eles propõe um tratamento utilizando *Webcam* voltado para acessibilidade, uso por idosos e até tratamento de segurança com foco na utilização da íris.

Em Ashraf (Ashraf et al., 2011), temos um tratamento de detecção de movimentos que utiliza o MATLAB, que é uma linguagem de alto nível com foco em processamento de imagens (MATLAB, 2015) para realizar a captura das imagens. Quando um movimento é encontrado, uma SMS (mensagem de texto via celular) é enviada ao número cadastrado no *software*. A proposta dos autores utiliza um telefone móvel com tecnologia de comunicação GSM ligada ao computador. O vídeo realizado pela *Webcam* é transformado em quadros e dois quadros são armazenados em *buffer*. A detecção de movimento ocorre quando é realizada a diferença desses dois quadros através de um auxílio de um "afinador" (tuning factor). Esse projeto possui, também, um circuito de alarme programado, celulares GSM e periféricos complementares, o que pode ser um incômodo para algumas pessoas comprarem, mesmo sendo baixo custo. Além de não possuir a implementação de detecção de faces.

[2] Imagem Integral é uma nova imagem gerada a partir de cálculos matemáticos com o objetivo de melhorar a quantidade de cálculos posteriores (VIOLA; JONES, 2001).

2.2 Detecção de Movimento

A detecção de movimento tem como objetivo identificar a região correspondente aos objetos que se movem em relação do resto da imagem. Numa imagem gravada por uma *webcam*, por exemplo, a detecção deve ocorrer quando houver alguma alteração nas imagens. Uma sequência estática de imagens não é detectada por esses algoritmos. A detecção de movimento pode ser encontrada não apenas a nível celular (ex: transmissões sinápticas), mas também a nível de computações feitas por pequenas redes neurais no campo da neurosciência computacional (BORST; EGELHAAF, 2015). Processos subsequentes como reconhecer rastreamentos e comportamentos, percorrer marcadores, entre outros processos, são dependentes disto (WANG et al., 2004).

Porém, um grande volume de vídeos digitais requer métodos eficientes de navegação, busca e visualização do seu conteúdo. Para recuperar informações de importância para o usuário, é necessário armazenar apenas os segmentos de vídeo relevantes. Com isso, a detecção de movimento faz com que apenas imagens de interesse sejam gravadas e, assim, economiza-se espaço em disco e recursos.

Sendo assim, pode-se relacionar tal conceito para aplicações mais específicas, como a proposta deste trabalho. Segundo Wang (WANG et al., 2004), para o caso dos sistemas de vigilância eletrônica, a deteçcão de movimento por meio de sinais de vídeo tem forte apelo ao uso de determinadas regiões na imagem.

2.3 Reconhecimento Facial

A face humana é o principal atributo anatômico pelo qual as pessoas são reconhecidas (NASCIMENTO; GONZAGA, 2005). Podemos utilizar desta informação para melhorar o sistema de segurança.

Segundo Yang, Kriegman e Ahuja (2002), os desafios associados com a detecção de rosto podem ser atribuídos aos seguintes fatores:

- Pose: as imagens do rosto humano podem variar dependendo de alguns fatores, como posição da câmera (reta ou inclinada);
- Componentes faciais: um rosto pode possuir barba, bigode e óculos podendo ocorrer uma variação grande, como diferenças entre óculos (alguns maiores, outros de diferentes cores), diferença entre barbas (cores, quantidade de pelos);
- A oclusão: as faces podem não ser completamente exposta, sendo bloqueada por outros objetos. Como exemplo, uma imagem com um grupo de pessoas ou animais, alguns rostos podem ficar apenas parcialmentes expostos;
- Condições de imagem: para que ocorra uma boa captura de rosto, alguns fatores como luz e tipo de câmera usada influenciam de forma direta nos resultados.

Contudo, existem algoritmos e bibliotecas na área de processamento de imagens que auxiliam no desenvolvimento desse tipo de sistema, como a OpenImaj (HARE; SAMAGOOEI; DUPPLAW, 2012) que será utilizada nesse trabalho.

2.4 Sistemas no Mercado

Existem alguns Sistemas no Mercado que usam a *WebCam* para as mais diversas tarefas. Apresenta-se nas seções seguintes uma revisão de algumas dessas ferramentas que estão diretamente ou indiretamente relacionadas a proposta desse trabalho.

2.4.1 CyberlinkYouCam 6

Cyberlink YouCam 6 é um *software* para o sistema operacional *Windows* com diversas funcionalidades como troca de mensagens, gravação de vídeos com efeitos, melhoramento de imagens, fotos panorama, adição de frames, corte e rotação. Além destas, encontra-se um módulo de segurança que é ativado pela detecção de movimento (CYBERLINK, 2015).

Nesse *software*, a vigilância implementada é para fins de entretenimento. A identificação de face e o envio de imagens por e-mail não estão implementadas.

2.4.2 *iSpy*

iSpy é um software que foi desenvolvido para a plataforma *Windows* com suporte para inúmeras câmeras. Possui detector de movimento e envia um *e-mail* ou SMS com as imagens para o usuário, caso algum desses eventos ocorra (ISPY. FEATURES, 2015).

Por ser implementado em *C#*, depende da plataforma exclusiva da Microsoft , *.NET*[3]. Sendo assim, não é multiplataforma. Embora seja *open source* com distribuição gratuita, o desenvolvedor espera por doações para continuar a dar suporte para seu desenvolvimento (ISPY. ABOUT, 2015).

2.4.3 Sighthound

Sighthound grava e armazena vídeos através de câmeras IP (que conectam utilizando a internet) e câmeras USB (conexão feita através de um cabo USB). Executa em *Mac* e *Windows* para *desktops* e *iOS* e *Android* para dispositivos *mobile*. Conta com recurso de gravação de vídeo através de regras específicas de detecção de movimentos, porém

[3] O .NET Framework é uma plataforma de desenvolvimento popular para a criação de aplicativos para Windows e inclui a linguagem de programação C# (MICROSOFT, 2015).

não existe versão para sistemas operacionais Linux (SIGHTHOUND, 2015). Nesse sistema também não está implementado algoritmo de detecção de face.

2.4.4 IP Camera Viewer

No *IP Camera Viewer*, é possível controlar no máximo quatro câmeras simultaneamente, sendo uma delas centralizada. Possui uma opção de gerenciamento de *layout* no qual permite a fiscalização das demais em tela única. É possível alterar a disposição da interface gráfica a fim de auxiliar as necessidades de segurança. Também oferece um *zoom* digital mesmo que não seja suportado pela câmera (IP CAMERA VIEWER, 2015).

O programa foi elaborado para a plataforma Windows e, apesar da abrangência de versões do sistema operacional (10/8/7/Vista/XP), não possui versão para Mac OS X e Linux.

3
Metodologia

3.1 Considerações Iniciais

Este trabalho tem como foco o baixo custo para o usuário final e a facilidade de implementação e programação a nível de desenvolvedor. Para isso, pesquisas foram realizadas a fim de descobrir quais bibliotecas, ambientes de desenvolvimento e sistema operacional cumprem com esses objetivos.

Para o início deste trabalho, foi utilizada a linguagem de programação orientada a objetos C++, sistema operacional com *Kernel Linux* (Distribuição *Linux Mint*), plataforma de desenvolvimento *NetBeans* e biblioteca *OpenCV*.

Segundo Bueno (2002), a linguagem C++ é uma das melhores linguagens de programação existentes por conseguir agrupar uma funcionalidade que envolve formulações altamente abstratas como classes, que permitem um trabalho de alto nível e formulações de baixo nível, como o uso de chamadas de interrupções que realizam tarefas sobre o hardware. Porém, as facilidades da linguagem de programação *Java* e a experiência adquirida ao decorrer da graduação foi um fator decisivo para a mudança, visto que o tempo gasto na implementação seria menor.

O Sistema Operacional considerado nesse trabalho foi o Linux, especificamente a distribuição *Linux Mint*. O *Linux Mint* é o terceiro sistema operacional mais usado em residências. Foi decisivo o uso por ser gratuito, *open source*, seguro e confiável (LINUX MINT, 2015). A escolha de um sistema operacional livre, ou seja, sob a licença GNU/GPL foi primordial no projeto, visto que um dos objetivos é alcançar uma grande economia de custos.

O *OpenCV* é uma biblioteca gratuita tanto para fins acadêmicos quanto para comerciais. É composta por interfaces em C, C++, Python e Java e é suportada nos

sistemas operacionais Windows, Linux, Mac OS, iOS e Android (OPENCV, 2015). Mesmo com sua biblioteca possuindo uma documentação consistente, a implementação desta juntamente com a linguagem Java no *Linux Mint* não obteve uma boa integração, visto que foi necessário a compilação manual da biblioteca e, durante este processo, vários erros ocorreram dificultando o uso. Por esta razão, foi realizada uma outra pesquisa em busca de uma nova biblioteca que possuísse funcionalidades iguais ou parecidas com àquelas fornecidas pela *OpenCV*. Dessa busca, foi encontrada a biblioteca *OpenIMAJ* que é usada neste trabalho.

Nas seções seguintes serão apresentadas as técnicas, ferramentas, linguagens e bibliotecas utilizadas no desenvolvimento desse trabalho.

3.2 *NetBeans IDE*

NetBeans é um ambiente de desenvolvimento duplamente licenciado sob as licenças *Common Development and Distribution* e GNU - *General Public License*. O Editor fornece modelos de código, dicas de codificação e ferramentas de refatoração. Oferece uma análise estática com integração especial com a ferramenta *FindBugs*, amplamente usada para a identificação e correção de problemas comuns em código *Java*. Além disso, o *NetBeansDebugger* permite inserir pontos de interrupção em seu código, avançar pelo código, executar métodos, obter telas e monitorar a execução durante sua ocorrência. É possível a instalação em todos os sistemas operacionais que possuem a JVM. (NETBEANS, 2015).

A escolha do *NetBeans* foi fundamental, visto que um ambiente de desenvolvimento tende a trazer benefícios para os projetos.

3.2 Linguagem *Java*

A linguagem *Java* foi escolhida para a programação. É uma linguagem orientada a objetos com todos os benefícios que um paradigma orientado a objeto pode oferecer, tais como simplicidade. A estrutura de programas e classes em *Java* segue a organização de linguagens tradicionais como *C*, mas sem elementos de programação complexos. Portabilidade: pode ser compilado em qualquer computador que possua a *JVM*. Além disso é gratuita, robusta e possui diversas bibliotecas prontas e seguras, possuindo ainda um recurso para utilização de múltiplos threads (HORSTMAN; CORNELL, 2010).

3.2.1 Paradigma Orientado a Objetos

É um paradigma criado a fim de facilitar e aproximar o programador do mundo real. Pelo fato do trabalho utilizar muito os recursos multimedia, a programação orientada a objetos tende a facilitar o cumprimento dos objetivos, visto que muitos objetos já possuem, de forma nativa da suíte Java, os métodos já prontos para o uso do programador.

Segundo Dall'oglio (2009), para aplicar a Programação Orientada a Objetos, primeiro é necessário entender o conceito de classes e objetos. Uma classe é uma estrutura que define um tipo de dados, podendo conter atributos (que representam variáveis) e também métodos (que representam funções e procedimentos) para manipular esses atributos. Um objeto contém exatamente a mesma estrutura e as propriedades de uma classe, no entando sua estrutura é dinâmica, seus atributos podem mudar de valor durante o tempo de execução do programa, o que torna o Paradigma Orientado a Objeto uma excelente técnica para aproximar o programador do mundo real.

3.2.2 *Thread*

Threads são atividades ou subprocessos, que são (ou parecem ser) executadas ao mesmo tempo por um programa. Quando o programa realiza uma tarefa principal e há uma necessidade de execução de outra tarefa, é possível disparar uma *thread* para começar esse trabalho sem e necessidade de bloquear a tarefa primária.

Em Java, a *thread* possui dois significados: uma instância da classe *java.lang.Thread* ou uma *thread* de execução. Uma instância de *Thread* é apenas um objeto e como qualquer objeto em *Java*, possui variáveis, métodos, reside e é eliminado. Uma *thread* de execução é um processo individual e possui sua própria pilha de chamadas. Mesmo se nenhuma nova *thread* for criada, elas estarão lá sendo executadas em segundo plano. Podem ser criadas herdando a classe *Thread* ou implementando a interface *Runnable* e toda ação inicia no método *run()*, que é o único método que deve ser subscrito da interface *Runnable*, mas a *thread* não se tornará uma *thread* de execução até que o método *start()* seja chamado. Durante a execução, caso algum evento ocorra, uma *thread* pode ficar bloqueada por um certo tempo. Para isso, pode-se chamar o método *sleep()*, que faz com que a *thread* fique suspensa por um determinado tempo em milissegundos. O método *yield()* bloqueia uma *thread* e permite que outra *thread* de mesma prioridade seja executada. O método *join()* de uma *thread* chama outra *thread* quando precisa parar sua execução para que a *thread* chamada seja executada. Uma *thread* morre no término da execução do método *run()* (HORSTMAN; CORNELL, 2010).

Neste trabalho, foi preciso utilizar os processos de forma paralela. A detecção movimento, detecção de um rosto, gravação de vídeo e exibição da imagem da *Webcam* para o usuário são atividades que precisam ser executadas ao mesmo tempo.

3.3 *Webcam Capture*

Na primeira tentativa de estabelecer uma conexão com a *Webcam*, a biblioteca inicialmente utilizada (*OpenCV*) continha arquivos de configurações responsáveis por manter aberta a conexão entre o hardware e o software, dificultando o processo da programação, o que gerou pesquisas em busca de novas bibliotecas.

Uma das bibliotecas encontradas foi a *Webcam Capture* que permite o uso diretamente da *Webcam* com base na linguagem *Java*, suportando múltiplas plataformas e diferentes arquiteturas. Também captura imagens de câmeras de baixa até alta resolução e possui classes prontas para a detecção de movimento (WEBCAM-CAPTURE, 2015).

No momento da construção do projeto, esta foi a biblioteca com maior facilidade de utilização encontrada, no qual abstrai-se a *Webcam* como um componente de *hardware* e traz os módulos prontos para que apenas ocorra a chamada pelo programa.

3.4 *OpenIMAJ*

O OpenIMAJ é um conjunto de ferramentas que foi, no início, fundada na União Européia por engenheiros, conselho de pesquisa de ciência física, conselho de pesquisa de artes e recursos humanos, pesquisa de artilharia e a BBC. A licença de uso é a *Creative Commons*, o que permite o uso, cópia e distribuição desde que os devidos créditos sejam fornecidos.

O *OpenIMAJ* também é responsável pela análise e geração de conteúdo multimídia. O *software* é estruturado em vários módulos que podem ser usados de forma independente. A manipulação de imagens usando somente os recursos e métodos disponíveis em *Java* não é muito intuitiva; por esta razão o *OpenIMAJ* possui

classes com métodos que tratam uma imagem como uma matriz de *pixels* (o que realmente é). A abstração da imagem na OpenIMAJ é melhor que a *ImageIO* do *Java*, facilitando, assim, o uso para o programador (HARE; SAMANGOOEI; DUPPLAW, 2012).

A escolha desta ferrameta se deu pela facilidade no manuseio dos objetos. Para detecção de faces, por exemplo, a biblioteca possui um detalhado modelo matemático que facilita a sua programação.

Várias outras bibliotecas foram testadas para a utilização com a linguagem Java, porém todas foram descartadas e a mais estável, prática e versátil foi o OpenIMAJ, responsável por grande parte da construção do código.

3.4.1 *Maven*

O *Apache Maven* é uma ferramenta de gerenciamento de projetos. Seu principal objetivo é permitir que o desenvolvedor alcance o maior aproveitamento em um curto período de tempo.

Permite a construção de um projeto por meio do POM, um arquivo XML, e de um conjunto de *plug-ins* que são compartilhados por todos os projetos que o utilize. A estrutura de diretórios de um projeto *Maven* possui as entradas *project home* que contém o pom.xml e todos os subdiretórios, *src/main/java* que contém o código-fonte *Java, src/main/resources* que contém os recursos para o projeto e *src/test/java* que contém as classes de teste. Quando um projeto é executado, é realizada uma busca pelo arquivo POM no diretório atual que obtem as informações de configurações necessárias para a execução (MAVEN, 2015).

O uso do *Maven* no trabalho ocorreu pelo fato do *OpenIMAJ* já possuir todas as configurações padrões completas como um sub-projeto, facilitando ainda mais para o programador. Com isso, apenas pequenos ajustes foram feitos a fim de adaptar a biblioteca ao *software* criado.

3.4.2 Reconhecimento Facial

A extração de características representativas de um conjunto de dados é uma tarefa complexa e exige modelos sofisticados (RAINA et al., 2003). Um conjunto de modelos matemáticos conhecido por resolver este problema é o Reconhecimento de Padrões da área de Inteligência Artificial. Porém, neste trabalho, é proposto uma maneira mais simples de resolver esse problema, visto que não é necessário o padrão de uma face, e sim se ela realmente está contida em uma imagem.

Neste trabalho, a proposta para o reconhecimento facial pode ser explicada, basicamente, como uma sub-janela situada no canto superior esquerdo que percorre toda a imagem, como ilustrado na Figura 1. O algoritmo de reconhecimento facial realiza o percurso dessa sub-imagem buscando um padrão de imagem que se encaixe com a face humana.

Figura 1: Possível posição de uma sub-janela.

Fonte: ROCHA (2012).

3.4.3 Haar Cascade Classifiers

O *Haar Cascade Classifiers* é um detector de faces baseado no algoritmo de *Viola-Jones*. O classificador vem com uma série de modelos pré-treinados para visualizações de rosto frontal e lateral, porém não detecta rostos não-verticais. (HARE; SAMAGOOEI;

DUPPLAW, 2012). A proposta dos classificadores (filtros) *Haar Cascade* é exclusivamente reconhecer a presença de um rosto na imagem através de cálculos que não necessite da complexidade de um modelo baseado em Inteligência Artificial.

Para que isso ocorra, são necessários filtros desenhados de tal forma a favorecer esse reconhecimento. As unidades básicas do método Viola-Jones são os filtros retangulares, que são divididos em três tipos, como ilustra a Figura 2: O filtro dois-retângulos é a diferença entre a soma de *pixels* dentro de duas regiões retangulares. As regiões possuem o mesmo tamanho e forma e são horizontais ou verticalmente adjacentes, como está representado nos itens A e B, da Figura 2. O filtro três-retângulos é a soma dos retângulos externos e subtrai-se com o retângulo do centro, como apresentado no item C. Por último, um filtro quatro-retângulos calcula a diferença entre os pares de retângulos na diagonal (item D). (VIOLA; JONES, 2001).

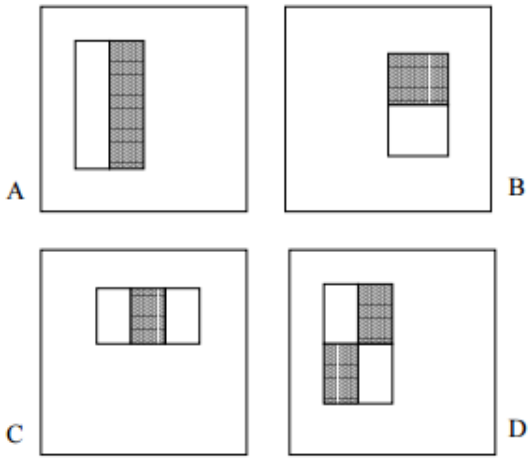

Figura 2: Possíveis configurações de um Filtro.

Fonte: VIOLA; JONES (2001).

De acordo com Viola e Jones (2001), os filtros retangulares podem ser calculados utilizando a representação da imagem integral definida pela equação:

$$\boldsymbol{ii}(\boldsymbol{x},\boldsymbol{y}) = \sum_{\boldsymbol{x}\leq\boldsymbol{x}',\boldsymbol{y}\leq\boldsymbol{y}'} \boldsymbol{i}(\boldsymbol{x}',\boldsymbol{y}') \qquad (1)$$

Em que os pares ordenados (x, y) representam os pixels da imagem, i (x', y') representa a imagem original e ii (x, y) representa a integral da imagem.

Com a imagem integral, é possível obter a avaliação para qualquer tipo de retângulo utilizando apenas quatro valores da imagem, ilustrados na Figura 3:

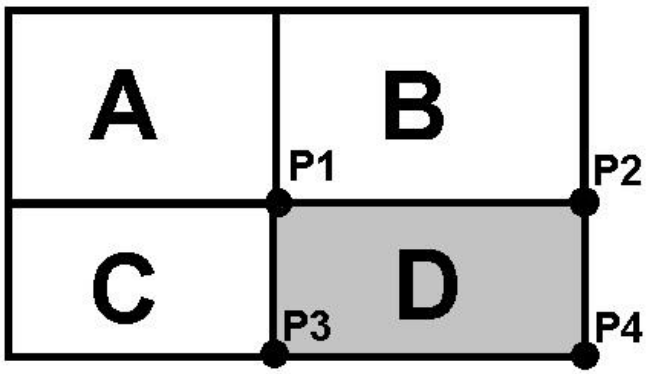

Figura 3: Aplicação dos cálculos na região D.

Por meio desta representação, calcula-se o valor do somatório dos *pixels* que ocupam determinada área retangular na imagem em complexidade constante, O(1)[4], pela equação:

$$D = \sum[P_1 + P_4 - (P_2 + P_3)] \quad (2)$$

Em que P_1, P_2, P_3 e P_4 representam o valor do pixel referente àquele par ordenado (x, y) da imagem.

Após a imagem integral realizar a soma dos *pixels* da área de interesse, ocorre o início de outra classificação para saber se contém as características de uma face. Essa classificação ocorre por meio de um cálculo que é baseado no algoritmo *AdaBoost* (FREUND; SCHAPIRE, 1996).

Os classificadores *Haar Cascade* abstraem a implementação dos cálculos baseados no *AdaBoost*, deixando ao cargo do programador apenas setar os

[4] Algoritmos cuja complexidade é da ordem de 1.

parâmetros da função que tem como objetivo aumentar ou diminuir a área visível em que se encontra uma face.

4
Sistema Desenvolvido

4.1 Considerações Iniciais

O sistema proposto nesse trabalho visa a gravação de vídeos a partir do momento em que a *webcam* detecta movimento na imagem. No instante em que um ser humano é reconhecido por meio de sua face, o *software* salva uma imagem no formato JPEG apenas do rosto do indivíduo. Com as imagens e vídeos salvos, o próximo passo do sistema é compactar em um arquivo com extensão .zip as fotos gravadas e enviar para um e-mail previamente cadastrado no programa.

O projeto foi desenvolvido utilizando a linguagem de programação *Java* cuja versão do JRE foi a 1.8.0_40-b26, tendo como base para implementação das Classes a biblioteca *OpenIMAJ* e os módulos fornecidos pela biblioteca *Webcam Capture*, descritas no Capítulo 3.

O gerenciador de projetos *Maven*, que também foi usado nesse trabalho, gera um arquivo pom.xml, que descreve todas as dependências e também contém instruções para o empacotamento em *.jar*, que irá conter todo o código do projeto. Essa funcionalidade está descrita no Capítulo 3.

4.2 Interface com o Usuário

Ao iniciar o programa, temos a busca pelas *webcams* conectadas ao sistema via USB. Após encontrada, o *software* insere a resolução da imagem, abre uma janela com a gravação do vídeo em tempo real e inicia o *buffer* para a recepção de imagens. Possui

um *ComboBox* na parte superior, designado a escolher todas as *webcams* que estejam plugadas e disponíveis no computador. No canto inferior esquerdo, temos o FPS (*frames per seconds*) que apresenta a quantidade de quadros (imagens) exibidos por segundo.

4.3 Detecção de Movimento

A Figura 4 mostra o momento em que o software é iniciado e, como exemplo, utilizando a *BisonCam*.

Figura 4: Tela inicial do *software*

O algoritmo responsável pela detecção de movimentos é também instanciado e um objeto faz a verificação da presença de movimentos através de cálculos que mostram diferenças entre as imagens já capturadas e o quadro atual. Após verificar, espera por 500 milissegundos até entrar em ação novamente. A escolha de 500 milissegundos para o descanso da *thread* responsável por esta ação se deu baseado

em testes empíricos, visto que esse tempo de espera não é alto a ponto de afetar a detecção de movimentos e também não é baixo ao ponto da thread permanecer em constante uso do processador.

```
public CameraMotion(Camera webcam){
    this.webcam = webcam;
    detector = new WebcamMotionDetector(webcam.getCamera());
    detector.setInterval(500);
    detector.addMotionListener(this);
    detector.start();
}
```

Figura 5: Construtor da classe responsável por instanciar um objeto detector

Como mostra a Figura 5, a detecção de movimento já está implementada na biblioteca *Webcam Capture*, restando ao programador setar os parâmetros de configurações de acordo com as necessidades de cada projeto. O conceito de *thread* também está abstrato no objeto, visto que é necessário seu uso por ser um cálculo realizado em paralelo em relação ao programa principal. A *thread* se inicia pela chamada do método "start()".

A classe *CameraMotion* implementa a interface *WebcamMotionListener*, que possui um método chamado *MotionDetected*, responsável por salvar o tempo do último movimento e iniciar a *thread* responsável pela gravação de vídeo, caso não exista um vídeo sendo gravado naquele momento.

4.4 Gravação do Vídeo

Quando um movimento é detectado, um novo vídeo é criado e a gravação começa.

A Figura 6 mostra que a gravação dos quadros inicia-se pelo método "webcam.setRecording(true)". Após isso, uma nova thread é criada e iniciada para que a gravação ocorra em paralelo, não sendo necessária a interrupção do programa principal.

```
public void motionDetected(WebcamMotionEvent wme) {
    webcam.setLastMovingTime(System.currentTimeMillis());
    if(!webcam.isRecording()){
        webcam.setRecording(true);
        System.out.println("Started recording...");
        t = new Thread(new CameraRecord(webcam));
        t.start();
    }
}
```

Figura 6: Método responsável pela gravação de vídeo

O vídeo gerado pela gravação utiliza codificação H.264[5].

No momento em que a gravação é requisitada, as imagens são armazenadas a uma taxa de 10 FPS no *buffer* e convertido no formato de vídeo com extensão .ts, que é estabelecida pela biblioteca como padrão. Possívelmente outras extensões são suportadas, porém não foram testadas.

Cada vídeo gravado possui um nome único, pois recebe a data definida no sistema (calendário do computador) como um dos componentes do nome do arquivo. Desta forma, os vídeos são sempre exclusivos impedindo que alguma gravação sobrescreva outra já existente.

4.5 Reconhecimento da Face

O objeto responsável por reconhecer faces é instanciado quando uma gravação de vídeo está em andamento. Ocorre, também, o uso de *thread*, visto a necessidade da execução em paralelo com a gravação de vídeo.

Ao instanciar um objeto *HaarCascadeDetector*, que é responsável pela detecção de faces, deve-se passar como parâmetro a quantidade mínima de *pixels* em que a face pode ser encontrada. Quanto maior a quantidade de *pixels*, maior será o espaço capturado ao redor da face detectada. A Figura 7 mostra a face detectada com o

[5] Padrão de compressão de vídeo conhecido por sua alta definição (H264, 2015).

parâmetro do objeto setado em 50 pixels. Esse valor se mostrou adequado ao propósito do projeto visto que, a face da imagem gerada é perfeitamente visível, ocupa pouco espaço no disco rígido e facilita o envio por e-mail.

Figura 7: Exemplos de faces capturadas.

A imagem resultante da captura é uma imagem em escala de cinza em que cada *pixel* é representado por um número em ponto-flutuante (HARE; SAMAGOOEI; DUPPLAW, 2012).

4.6 Armazenamento das Faces

O processo de armazenamento das faces pela detecção do *software* é inserido de forma individual em uma lista encadeada. Pelo fato de que, em uma única foto, é possível que exista mais de um rosto, a lista se mostrou bem adaptada para esse uso.

Para cada face reconhecida, é criado um arquivo com nome diferente utilizando a data do sistema pela biblioteca *Calendar* do próprio *Java*, conforme explicado anteriormente. Assim, garante-se a consistência em disco do arquivo relacionado a cada face gravada.

4.7 Compactação e Envio por E-mail

Para mais segurança e praticidade para o usuário, o *software* possui um módulo de compactação e envio por e-mail para que as fotos do rosto do suspeito não precisem ficar armazenada no computador local.

O algoritmo consiste em, ao capturar 50 fotos, processar a compactação das faces em um arquivo .zip e em seguida, enviar para o e-mail configurado. A escolha do envio de exatos 50 fotos foi para fins de testes, mas o programa se mostrou adaptável a números maiores ou menores que 50, ficando a cargo do usuário escolher a quantidade de fotos deseja receber por e-mail.

Após a confirmação de envio, as imagens das faces e o arquivo compactado são deletados da memória secundária, restando apenas uma cópia mantida no e-mail de saída.

5
Resultados

O programa gera, como resultado, fotos das faces detectadas e vídeos gravados resultante da percepção de movimentos, que são armazenados no computador do usuário pela divisão de pastas definidas de acordo com a data setada no computador naquele instante, como mostra um exemplo na Figura 8.

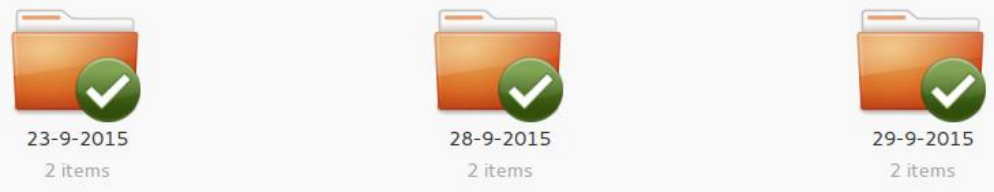

Figura 8: Arquivos resultantes armazenados em pastas.

Dentro de cada pasta referente ao dia da gravação, é criado uma sub-divisão de novas pastas com o nome "fotos" e "vídeos" a fim de melhor organizar cada tipo de arquivo, como mostrado na Figura 9.

Figura 9: Pastas contendo as fotos e vídeos gerados pelo *software*

A Figura 10 mostra um exemplo dos arquivos dentro das pastas, neste caso, a pasta "fotos".

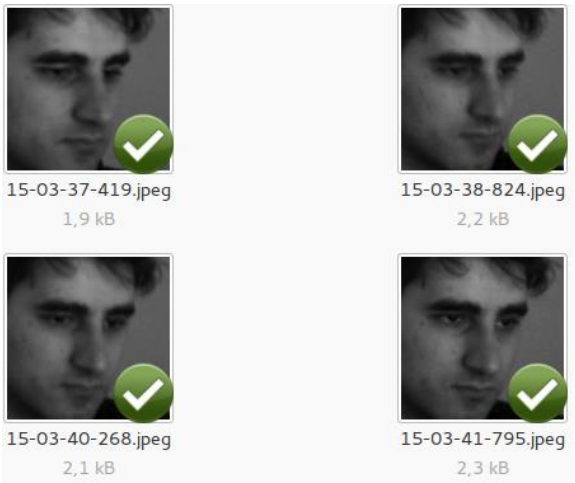

Figura 10: Organização das fotos dentro da pasta "fotos".

O programa se mostou eficiente em relação ao armazenamento local, visto que as fotos são armazenadas em disco rígido de forma temporária, restando apenas alocar espaço para guardar os vídeos.

Pelos testes realizados, o algoritmo que calcula as faces se mostrou eficiente em ambientes iluminados com luz natural e com luz exclusivamente artificial, se restringindo apenas a ambientes escuros. A detecção dos movimentos também foi eficaz em ambas as situações de luminosidade. A gravação de vídeo também apresentou um armazenamento satisfatório, mostrando que, com uma hora direta de gravação, apenas 200 megabytes foram alocados em disco.

A webcam usada no projeto foi a *BisonCam NB Pro*, um modelo simples fabricado pela *Acer* cuja resolução de imagem é 640x480 pixels, mostrando que a eficiência dos algoritmos é funcional mesmo nos modelos simples. Porém, acredita-se que melhores resultados podem ser alcançados com modelos superiores.

Em relação às fotos capturadas pelo programa, são logo deletadas após a compactação e o envio, como mencionado no Capítulo 4. Pelo fato das imagens armazenadas terem um tamanho relativamente pequeno em relação ao permitido no envio por um único e-mail, a junção de 50 num arquivo .zip não afetou o limite máximo dos principais servidores de *e-mails* testados, como *Gmail* e *Hotmail*. Sendo assim, o usuário pode confiar no arquivo anexo que será enviado.

6
Conclusões e Trabalhos Futuros

O objetivo desde trabalho de promover um sistema de vigilância útil, acessível à baixa renda e integrável em sistemas operacionais *Linux* foi cumprido. Apesar do foco e plataforma de desenvolvimento ser o *Linux Mint*, foi constatado perfeito funcionamento no sistema operacional mais usado do mundo, o *Windows*, graças à multiplataforma oferecida pela JVM. Os códigos gerados pelo trabalho foram simples e objetivos, facilitando futuras implementações por diferentes programadores.

Embora a nível do usuário final nenhuma configuração a respeito do *software* possa ser feita, é possível várias implementações futuras com muitas melhorias. Como exemplo, escolha da resolução (dentre as suportadas pela *Webcam* em execução), para uma situação em que se deseja apenas um pequeno quadro em exibição no monitor do usuário. Adição de mais destinatários de *e-mail*, para algum cenário em que mais de um envolvido no negócio recebesse instantaneamente o envio da mensagem pelo *software*. Envio por e-mail dos vídeos gerados, a fim de complementar o uso dos *e-mails*. Criação de uma funcionalidade que visaria a não gravação de imagens cuja expressão facial já exista salva em disco ou são parecidas, a fim de economizar mais espaço de armazenamento e melhorar o desempenho do reconhecimento do programa. E como fator opcional, deixar como escolha do usuário alguma opção que não lhe interessar, podendo ser desabilitada.

Uma análise técnica sobre as extensões das imagens e dos vídeos poderiam ser estudadas e comparadas, assim como o estudo de provas matemáticas a fim de otimizar os cálculos e o sistema como um todo.

Uma outra modificação do *software* que visa exclusivamente os vídeos, atualmente sendo salvos em pastas locais, poderiam ser gravados em disco virtual,

distribuindo melhor os resultados em várias outras plataformas ou dispositivos, como celulares. Serviços tais como o *Dropbox*, *OneDrive* ou *Google Drive* são gratuitos e oferecem esse tipo de integração.

7
Referências Bibliográficas

BRASIL. Ministério da Justiça. Secretaria Nacional de Segurança Pública. **Sistema Nacional de Informações de Segurança Pública**. 2013. Disponível em: <https://www.sinesp.gov.br/estatisticas-publicas>. Acesso em: 08 jul. 2015.

BRASIL. Governo do Estado de São Paulo. Secretaria da Segurança Pública. **Produtividade Policial.** 2015. Disponível em: <http://www.ssp.sp.gov.br/novaestatistica/Pesquisa.aspx>. Acesso em 08 jul. 2015.

LINUX MINT. ***About***. 2015. Disponível em: <http://www.linuxmint.com/about.php>. Acesso em 09 jul. 2015.

OPENCV. ***Open Source Computer Vision***. 2015. Disponível em: <http://opencv.org>. Acesso em 09 jul. 2015.

SIGHTHOUND. ***Frequently Asked Questions***. 2015. Disponível em: <https://www.sighthound.com/frequently-asked-questions>. Acesso em 08 jul. 2015.

MEDEIROS, H. Téchne. **Casa do Futuro**. 2009. Disponível em: < http://techne.pini.com.br/engenharia-civil/143/casa-do-futuro-286568-1.aspx>. Acesso em 08 jul. 2015.

ROCHA, A. **Localização de Faces**. 2012. Disponível em: <http://www.ic.unicamp.br/~rocha/teaching/2013s1/mc851/aulas/additional-material-viola-jones.pdf>. Acesso em 30 set. 2015.

RAINA R.; SHEN Y.; NG, A.Y.; MCCALLUM, A. ***Classification with hybrid generative/discriminative models***. 2003. *Advances in Neural Information Processing Systems*.

NERES, W.F. **Segurança Residencial: Monitoramento Visual do Ambiente via Internet com Alarme via E-mail**, 2007, 66f. Trabalho de Conclusão de Curso (Engenharia da Computação) - Centro Universitário de Brasília, Brasília, 2007

GUADALUPE, M. B. **Preditores da compra de alarme residencial: variáveis de cenário e da história de aprendizagem**, 2014, 58f, Dissertação (Ciências do Comportamento) - Universidade de Brasília, 2014

KUROSE. J. F.; ROSS K.W. **Redes de Computadores**. 5. ed. São Paulo: Pearson, 2011

BORST, A.; EGELHAAF, M. ***Principles of visual motion detection***, Elsevier Science, v. 12, 1989.

MICROSOFT. Visual Studio. ***.Net Framework***. 2015. Disponível em: < https://msdn.microsoft.com/pt-br/vstudio/aa496123.aspx>. Acesso em 09 jul. 2015

CYBERLINK. YouCam 6 Deluxe. ***Features***.2015. Disponível em: <http://www.cyberlink.com/products/youcam/features_en_US.html?&r=1> Acessoem: 09 jul. 2015.

ISPY. ***Features***. 2015. Disponível em: < http://www.ispyconnect.com/features.aspx>. Acesso em: 09 jul. 2015.

ISPY. ***About***. 2015. Disponível em: <http://www.ispyconnect.com/about.aspx>. Acesso em: 09 jul. 2015.

WEBCAM-CAPTURE. ***Webcam Capture API***. Disponível em: <http://webcam-capture.sarxos.pl/>. Acesso em: 10 jul. 2015.

H264. ***What is H.264***, 2015. Disponível em: <http://www.h264encoder.com>. Acesso em 10 jul. 2015.

BUENO, A. D. **Apostila de Programação Orientada a Objetos em C++**. Universidade Federal de Santa Catarina - Laboratório de Meios Porosos e Propriedades Termofísicas e Núcleo de Pesquisa em Construção, v. 0.4, p. 60, 2002

HARE, J.; SAMAGOOEI, S.; DUPPLAW, D. ***The Open Image Tutorial***.1.3.1. Ed. The University Of Southampton, 2012.

HORSTMAN, C. H.; CORNELL, G. ***Core Java***.8 ed., v. 1, Pearson Prentice Hall, 2010.

IP CAMERA VIEWER. Produtos. **Vigilância de vídeos**. Disponível em: <http://www.deskshare.com/lang/po/ip-camera-viewer.aspx>. Acesso em: 09 jul.2015.

DALL'OGLIO, P. **PHP, Programando com Orientação a Objetos**. 2. ed. São Paulo: Novatec, 2009

MATTOS, G. I.; ANDREATTA, V. **Protótipo de Aplicativo para Monitoramento de Ambientes Internos através de Detecção de Movimento em Sequência de Vídeo**. Departamento Acadêmico de Informática, Universidade Federal do Paraná, Curitiba, Brasil, 2010.

MAVEN. ***Welcome to Apache Maven***. 2015. Disponível em: <http://maven.apache.org>. Acesso em: 10 jul. 2015.

MIGUEL, P. R. G. **Sistema Computacional de Alarme e Vigilância Via IP**. Núcleo de ciências Exatas e Tecnológicas, Centro Universitário Positivo, Curitiba, 2007.

NASCIMENTO, A. V.; GONZAGA, A. Detecção de Faces Humanas em Imagens Digitas: Um Algoritmo Baseado Em Lógica Nebulosa. In: WVC, 10, 2005**Anais do I Workshop de Visão Computacional**, São Paulo, 2005 p. 96-99.

NETBEANS. Disponível em: <https://netbeans.org>. Acesso em: 09 jul. 2015.

MATLAB. Disponível em: < http://www.mathworks.com/products/matlab>. Acesso em: 14 nov. 2015.

VIOLA, P.; JONES, M. ***Rapid Object Detection using a Boosted Cascade of Simple Features***.Conference On Computer Vision and Pattern Recognition. 2001

WANG, L.; HU, W.; TAN, T.F.; MAYBANK, S. ***A Survey on Visual Surveillence of Object Motion and Behaviors***. IEEE Transactions on Systems, Man and Cybernetics, v. 3, p. 334-352, 2004.

LEE, C.G.; LIN YU-TZU; LIN RUEI-YAN; LIN YU-CHIH; ***Real-time eye-gaze estimation using a low-resolution webcam.*** Springer Science Business Media, 2012.

YANG, M; KRIEGMAN D. J.; AHUJA, N. ***Detecting Faces In Images: A Survey***. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 24. 2002.

FREUND, Y.; SCHAPIRE R.E. A ***Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.*** 1996 J*ournal of computer and system sciences* 55, 119-139, 1997.