colloquium exactarum

DETECÇÃO DE INTRUSÃO COM RECONHECIMENTO FACIAL EM IMAGENS GERADAS POR CÂMERAS DE SEGURANÇA

INTRUSION DETECTION WITH FACIAL RECOGNITION ON IMAGES GENERATED BY SECURITY CAMERAS

Bruno Tsutsumi Kuroiwa[1], Silvio Carro[1]

[1]Faculdade de Informática – FIPP, Universidade do Oeste Paulista – UNOESTE
E-mail: issao25@msn.com, silvio@unoeste.br

RESUMO - Os sistemas de segurança com câmeras de monitoramento comerciais surgiram na década de 70 e de lá para cá seu uso vem se tornando cada vez mais comum. Eles evoluíram desde sua concepção, com funções como gravações das imagens, câmeras móveis e rotativas, sensores de movimento, e nos últimos anos, auxílio de computadores. O presente artigo faz a proposta de um framework que fornece funcionalidades de alto nível, utilizando recursos disponíveis de câmeras de vigilância do tipo IP e técnicas em visão computacional, dentre elas a delimitação de regiões críticas, detecção e reconhecimento de faces e geração de logs de intrusão.
Palavras-chave: Reconhecimento Facial; Detecção de Intrusão; Câmeras de Segurança; Framework.

ABSTRACT - Commercial security systems with surveillance cameras emerged in the 70s, and their use is becoming more and more common since then. They evolved a lot since their creation, with functions as image recording, mobile and rotating cameras, movement sensors and, in recent years, with the help of computers. The present paper proposes a framework that provides high-level functionalities, using available resources of IP security cameras and some computer vision techniques, such as delimiting critical areas, facial detection and recognition and intrusion log generation.
Keywords: Facial Recognition; Intrusion Detection; Security Cameras; Framework.

Recebido em: 05/05/2015
Revisado em: 22/05/2015
Aprovado em: 18/06/2015

1 INTRODUÇÃO

Nos últimos anos, com a crescente busca por mais segurança auxiliada pela tecnologia, houve um expressivo aumento na popularidade dos sistemas de monitoramento por câmeras. Atualmente, porém, muitos deles são limitados a gravar as imagens continuamente, ou, no máximo, gravar e recordar apenas momentos com movimento nas imagens. Funcionalidades avançadas como marcação de regiões especiais, associar contexto e significado para pontos estratégicos ou identificar pessoas nas imagens fazem parte apenas de sistemas bem específicos e de preços muito elevados.

O objetivo principal do presente estudo é sugerir soluções na forma de um framework que visam potencializar os recursos pertinentes à vigilância por câmera digital. Ao aplicar técnicas na área de processamento digital de imagens e reconhecimento facial, auxiliado por tecnologias open source, serão criadas soluções para o controle de intrusão, reconhecimento facial dos intrusos e recursos para alertas de intrusão, tudo em um framework, para possível uso em outras aplicações.

Para uma melhor compreensão do trabalho, ele está dividido neste artigo da seguinte maneira: a seção 2, onde estão explicados brevemente a metodologia empregada, os materiais utilizados e as etapas da pesquisa; na seção 3 estão explicados alguns conceitos básicos utilizados, necessários para melhor entendimento do trabalho; logo após, a seção 4, com um detalhamento das etapas mais importantes no processo de estudo das ferramentas e da criação do framework; e finalmente, na seção 5, as considerações finais do autor, uma conclusão do trabalho e possíveis trabalhos futuros.

2 METODOLOGIA

A metodologia do projeto é de natureza aplicada, e as pesquisas do tipo experimental.

O projeto requisitou os seguintes materiais para sua realização: uma única câmera de segurança IP para testes e um computador com acesso à rede, além de diversos materiais digitais para auxílio no desenvolvimento, como bibliotecas do Java, artigos científicos e exemplos de aplicações com OpenCV.

O projeto em si foi dividido em três etapas: primeiro, uma extensa revisão bibliográfica sobre todos os conceitos que seriam utilizados na pesquisa; em seguida, trabalhos em cima de protótipos para aplicações e testes de conceitos estudados e, por último, o desenvolvimento do framework com suas funções definitivas, prontas para

uso, juntamente com um pequeno projeto para demonstração de seu uso.

3 CONCEITOS BÁSICOS

3.1 Tecnologias utilizadas

No decorrer do projeto foram utilizadas algumas bibliotecas, todas gratuitas, para a pesquisa, implementação e os testes. Todas as etapas do projeto pertinentes à programação foram realizadas utilizando linguagem Java.

3.1.1 OpenCV e JavaCV

Os processos que envolvem detecção e reconhecimento facial foram realizadas utilizando-se as bibliotecas OpenCV e JavaCV.

O OpenCV é uma biblioteca em linguagem C, criada no final da década de 90, voltada para visão computacional e aprendizado de máquinas. Atualmente ela é open source, podendo ser utilizada e modificada com restrições mínimas, seja para fins acadêmicos ou comerciais. Para o projeto foi utilizada a versão 2.4.9 do OpenCV.

O JavaCV é uma biblioteca de classes *wrapper*, que serve como uma interface para os métodos do OpenCV, realizando as conversões necessárias entre as linguagens C e Java.

3.1.2 XStream

O XStream é uma biblioteca gratuita que realiza a serialização de objetos para arquivos XML e vice-versa, criada em 2003 por Joe Walnes.

Neste projeto, o XStream é utilizado para realizar a gravação e leitura dos cadastros das pessoas e todas as informações das câmeras utilizadas e para a geração dos logs de intrusão. Todos esses processos de leitura e gravação serão explicados posteriormente, na descrição do framework.

3.2 CCTV

Os CCTV (Closed Circuit Televison, ou circuitos fechados de televisão) são ferramentas vitais para a segurança dos mais diversos locais. Eles são assim denominados devido à sua principal característica: as câmeras de vídeo destes circuitos transmitem as imagens apenas para um número limitado e específico de monitores. O primeiro CCTV registrado foi utilizado pela Alemanha, em 1942, instalado pela Siemens AG para observar o lançamento de mísseis V-2. Na década de 40 os Estados Unidos também se utilizam de CCTV para observar e estudar os efeitos de bombas atômicas, de uma distância segura; até hoje são utilizados no lançamento de mísseis e foguetes.

Na década de 60 começou a instalação de CCTVs em locais públicos, na Grã-Bretanha, para monitorar o tráfego de veículos e visualizar locais de acidentes de trânsito. Em 1968, devido ao aumento das taxas de criminalidade em Nova York, o

governo dos Estados Unidos mandou instalar nas ruas várias câmeras de vídeo; o uso dos CCTV foi um sucesso no combate ao crime, o que levou o próprio governo a fazer uso das tecnologias emergentes no combate aos diversos tipos de crimes (SILVA, 2009).

Nas décadas de 70 e 80, com a invenção do videocassete para uma recordação automática das imagens, muitas lojas, bancos e empresas de segurança começam a optar pelo uso dos CCTV como um método de prevenir e recordar crimes. É até hoje a aplicação mais comum destes circuitos, conhecidos geralmente por sistemas de monitoramento ou sistemas de segurança; é possível encontrar estes sistemas na maioria dos locais, desde simples lojas de conveniência, estações de trem, aeroportos, até os prédios mais luxuosos.

Nas últimas décadas os sistemas de segurança que fazem uso dos CCTV evoluíram bastante, acompanhando o grande salto tecnológico do último século. Como já citado anteriormente, a invenção do videocassete permitiu a gravação automática das imagens, o que dispensou a presença de uma pessoa o tempo todo assistindo aos monitores. O método multiplex de transmissão de dados permitiu que várias câmeras de vídeo fossem utilizadas ao mesmo tempo, transmitindo as imagens num mesmo canal. A criação do computador foi um importante passo na história, e ele atualmente é essencial para a contínua evolução na área. Recursos computacionais como controle do movimento da câmera, diversas análises em tempo real das imagens, utilização e interpretação de informações provenientes de sensores de calor, movimento, infravermelho, aumentam o poder e a importância dos sistemas de segurança.

3.3 Detecção e Reconhecimento de Faces

O rosto de uma pessoa é o principal fator utilizado para reconhecê-la, no dia-a-dia. Para o ser humano, reconhecer um rosto familiar é uma tarefa trivial, realizada constantemente; nós somos capazes de guardar uma quantidade imensa de rostos em nossa memória, e mesmo assim reconhecer a maioria delas em uma pessoa em questão de décimos de segundos.

Geralmente, ao planejar um algoritmo para a simulação de uma atividade humana, tentamos simular o nosso raciocínio, como pensamos ao realizar a tarefa. Porém, nem mesmo a ciência consegue explicar ainda como é o mecanismo utilizado pelos nossos cérebros para o reconhecimento de uma face (SILVA; SANTA ROSA, 2004), e por isso essa área ainda enfrenta alguns problemas, e tantas aproximações diferentes existem.

A técnica de reconhecimento facial utilizando computadores é relativamente nova, e existem poucos softwares que

trabalham com ela. Como muitos recursos de análise biométrica, existem algumas barreiras que limitam o real poder e utilidade da técnica. Por exemplo, sistemas de reconhecimento de voz não são indicados para áreas com muito barulho, analisadores de mãos e digitais não funcionam perfeitamente em lugares com bastante poeira ou oleosidade e, no caso do reconhecimento facial, elementos como luminosidade, mudanças nas características faciais, alguns acessórios e principalmente o ângulo de captura da imagem podem afetar negativamente o sistema (NORMAN, 2011).

Apesar das limitações, o estudo e aprimoramentos dos algoritmos de reconhecimento facial continuam, pois é uma técnica promissora que tem potencial, principalmente como uma verificação biométrica de segurança, que hoje auxilia nas maneiras convencionais de verificação como cartões e senhas, mas pode um dia substituir completamente esses métodos. Por exemplo, já existem computadores que só permanecem ligados caso um usuário com permissão fique na frente do monitor, e caso ele saia ou outra pessoa tente usar o computador, ele é desligado. Alguns bancos dos EUA se utilizam da técnica de reconhecimento facial como uma camada a mais de segurança em seus caixas eletrônicos, o que ajudaria muito países como o Brasil, onde se ocorrem tantos roubos e clonagem de cartões de crédito (BONSOR; JOHNSON, 2011).

Existem diversos algoritmos para detecção e reconhecimento de faces, e cada um analise e compara as faces de uma maneira diferente. Alguns trabalham comparando uma foto com outra (1:1) apenas para confirmação de identidade, outros são melhores para comparar com múltiplas imagens (1:N), otimizados na velocidade mas ao custo de uma menor precisão; podem utilizar modelos 3D para captura de profundidade e curvaturas no desenho da face; muitos algoritmos se utilizam dos pontos de características faciais (chamados pontos nodais) e comparam suas posições e distâncias entre elas (AMORIM; BERCHT, 2009). Geralmente não é possível obter duas imagens idênticas para se comparar, então o algoritmo deve também analisar as faces levando em conta um padrão de tolerância de erros.

Algumas técnicas podem ser aplicadas para aumentar a precisão da análise. Alguns softwares existentes no mercado consideram, através de uma imagem-amostra da pele da pessoa, características como rugas, poros e inclusive a textura geral da pele. Softwares que se utilizam de algoritmos para tratamento de iluminação e sombras também podem ajudar na nitidez das faces.

3.3.1 Algoritmos de reconhecimento facial

O OpenCV trabalha atualmente com três algoritmos de reconhecimento facial: Eigenfaces, Fisherfaces e LBPH (Local Binary Patterns Histograms).

Os métodos utilizando Eigenfaces e Fisherfaces são semelhantes: trabalham com matrizes para representação dos pixels das imagens, utilizam-se de uma imagem média criada a partir de todas as imagens usadas no treinamento e detectam nas imagens os pontos em que elas mais variam, ou seja, as informações que mais se diferem de uma pessoa para outra, para melhorar o processamento (ao invés de analisar toda a imagem). Para um bom funcionamento e uma boa taxa de acertos no reconhecimento, esses métodos requerem muitas imagens na fase de treinamento do algoritmo, além de serem mais notavelmente afetados por fatores externos, principalmente as diferenças de iluminação.

Para o reconhecimento facial trabalhado no projeto foi utilizado o algoritmo LBPH; ele é capaz de reconhecer faces mesmo com um número baixo de imagens treinadas, algo que em situações reais é muito comum: o framework pode ser usado por pequenas empresas, onde o número de funcionários cadastrados e suas respectivas fotos são baixos.

O LBPH funciona da seguinte maneira: a imagem é dividida em várias regiões, denominadas janelas. Cada pixel da imagem é comparado com seus vizinhos, e associado a esse vizinho o bit 1 caso ele seja igual ou maior, e 0 caso contrário; ao fim desse processo, os resultados podem ser concatenados em um número binário de 8 bits, chamado Local Binary Pattern (LBP) como mostrado na Figura 1; para cada uma das janelas é criado um histograma com esses valores binários. É este histograma que representa esta janela na comparação com outras imagens (MATURANA, 2009).

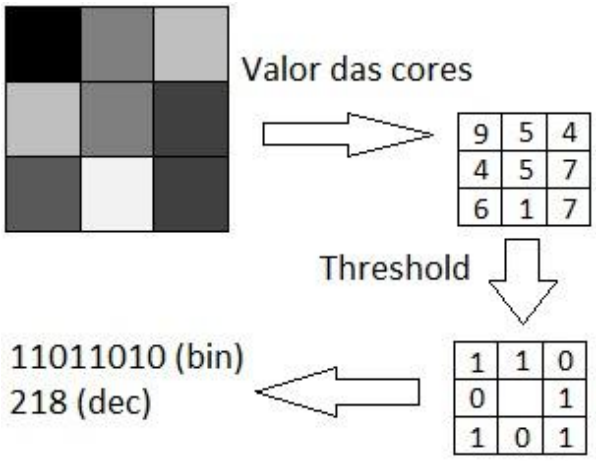

Figura 1. Comparação do pixel (tons de cinza)

4 DESENVOLVIMENTO

4.1 Framework RegionIntruder

Uma das etapas do projeto é o desenvolvimento do framework RegionIntruder, que deve ser capaz de realizar, através do uso das classes embutidas em seu framework e de algumas funções, os seguintes processos básicos: cadastro de

pessoa, cadastro de câmeras, delimitação de regiões de interesse em cada câmera, e principalmente a monitoração, que inclui o reconhecimento das pessoas que aparecem nas regiões delimitadas e a geração de logs de intrusão quando necessário, se solicitado.

4.1.1 Modelo de Classes

O modelo de classes do framework está apresentado na figura 2.

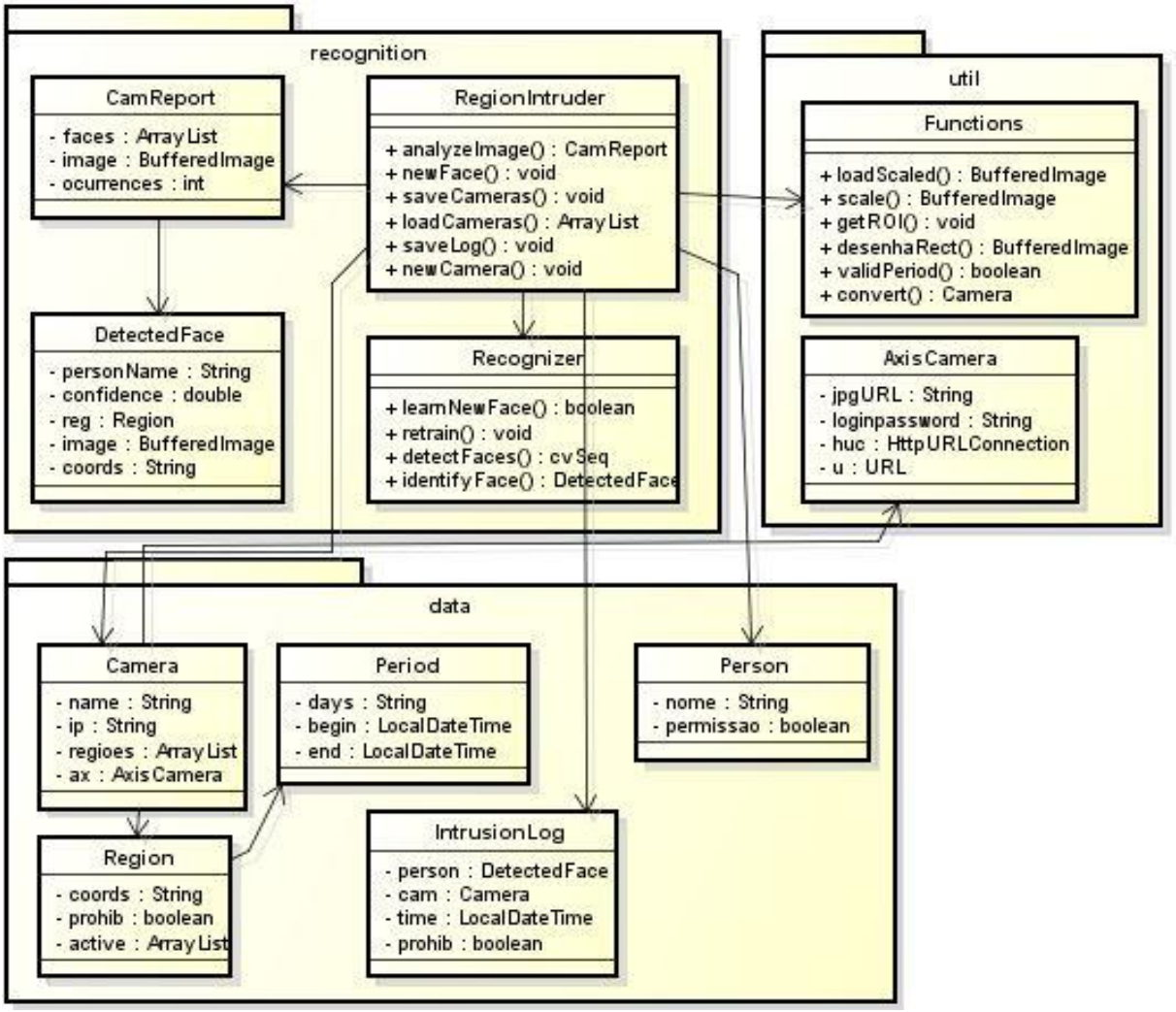

Figura 2. Modelo de classes do Framework

As classes do framework estão divididas em três pacotes:

- No pacote “Data” estão localizadas as classes bases do projeto, que representam as entidades envolvidas em todo o processo;
- No pacote “Recognition” estão as classes que trabalham com o processo de detecção e reconhecimento facial;
- O pacote “Util” contém algumas classes auxiliares e utilitárias.

Na Tabela 1 estão explicados o conteúdo básico e algumas funções de cada classe.

Além disso, o framework trabalha com um conjunto de diretórios pré-definido, sendo a raiz desses diretórios determinado pelo usuário, se ele desejar. A hierarquia dos diretórios está apresentada na figura 3.

Tabela 1. Detalhamento das classes do Framework

Classes	Detalhes
Person	Representa uma pessoa; para o projeto, só são necessários o nome, se ele tem autorização e pelo menos uma foto.
Camera	Representa uma câmera; possui informações como o IP da câmera e um nome, além de um ArrayList com as regiões de interesse.
Region	Representa uma região de uma câmera; além das coordenadas dessa região, é preciso saber se é controlada ou proibida, além dos períodos que ela funciona.
Period	Representa um período de tempo; associado à uma região.
IntrusionLog	Representa um log de intrusão, com os seguintes campos: a câmera correspondente, o tipo da região, a face encontrada e o horário.
RegionIntruder	Classe principal do framework, é através dela que o usuário consegue cadastrar e alterar pessoas e câmeras, ler e gravar nos respectivos XML e chamar a função analyzeImage(), que por usa vez utiliza a classe Recognizer e retorna um CamReport.
Recognizer	Responsável por todos os processos de detecção e reconhecimento de faces, desde o tratamento das imagens até a identificação dos possíveis rostos.
CamReport	Relatório da câmera, retorno da função analyzeImage(). Possui as seguintes informações: câmera, imagem da câmera, uma lista com todas as faces encontradas (ver classe DetectedFace) e o número de intrusões ocorridas.
DetectedFace	Representa um rosto detectado em uma imagem; possui informações como: região da câmera, imagem do rosto, nome da pessoa, um valor de confiança entre o rosto e a pessoa, e as coordenadas deste rosto encontrado.
Functions	Classe que possui algumas funções auxiliares e utilitárias para o projeto, como cópia de imagens, verificação de períodos de atividade de regiões, conversores para leitura e escrita em XML, etc.
AxisCamera	Representa a câmera utilizada no projeto, necessária para a conexão na rede e consequentemente a captura de suas imagens.

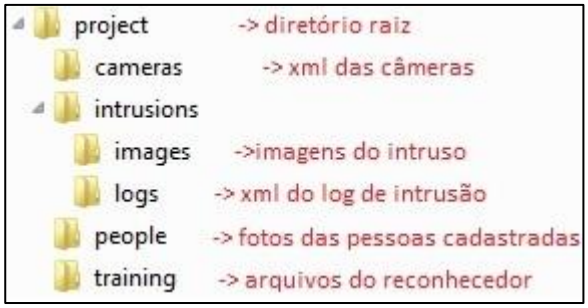

Figura 3. Hierarquia de Diretórios.

4.1.2 Cadastros de pessoas e câmeras

O cadastro de pessoas e câmeras geralmente é realizado em sua maior parte antes da execução do programa, e depois periodicamente, de acordo com a necessidade.

Para o funcionamento do framework, é necessário para uma pessoa no mínimo um nome, uma foto, e saber se ela tem autorização para entrar em áreas controladas; para a câmera, apenas o IP dela já basta.

Todos os cadastros de novas pessoas ou câmeras seguem uma mesma rotina, como visualizados nas figuras 4 e 5.

- Um objeto da classe Person ou Camera é criado e instanciado.
- O usuário tem a liberdade de escolher a maneira que esses objetos serão carregados. Na ferramenta foram utilizadas simples janelas de diálogo para inserção das informações
- Os objetos são passados para a classe RegionIntruder e são persistidos, todas as alterações sobrescrevendo o arquivo original, se este existir.

```
RegionIntruder reg = new RegionIntruder();
Person p = new Person("Nome da Pessoa");
p.setPermission(true);
reg.newFace(p,imagemDaPessoa);
reg.newFace(p,outraImagem);
```

Figura 4. Cadastro de Pessoa

```
RegionIntruder reg = new RegionIntruder();
Camera cam = new Camera();
cam.setIp("IP da câmera");
cam.setName("Nome para a câmera");
reg.newCamera(cam);
reg.saveCameras();
```

Figura 5. Cadastro de Câmera

4.1.3 Delimitação de regiões

Nem sempre todo o local vigiado por uma câmera é importante; muitas vezes apenas um ou alguns locais são de importância. Este framework possibilita a demarcação de áreas de interesse para cada câmera. Além disso, cada região pode ser definida como controlada ou proibida. Regiões controladas permitem apenas pessoas autorizadas nelas, enquanto regiões proibidas não aceitam ninguém.

Uma câmera pode ter novas regiões delimitadas, bem como alterar e remover regiões existentes. O usuário decide a maneira que ele deseja que seu programa realize as mudanças na câmera. O processo, na ferramenta, de criação de novas regiões, por exemplo, ocorre da seguinte forma:

- Com a câmera carregada no programa, o usuário clica em dois pontos no painel central, correspondendo ao canto superior esquerdo e ao inferior direito da região desejada, respectivamente.
- Na janela que se abre, o usuário pode configurar esta nova região: seu tipo (controlada ou proibida) e os períodos em que ela deve estar ativa, como mostrado na figura 6. Caso nenhum período seja adicionado, esta região é considerada inativa.
- Para salvar essa câmera com a nova região, o usuário a envia para a classe RegionIntruder, que irá persisti-la como se fosse um nova câmera. Utiliza-se o mesmo método para criar uma nova câmera e para atualizar uma.

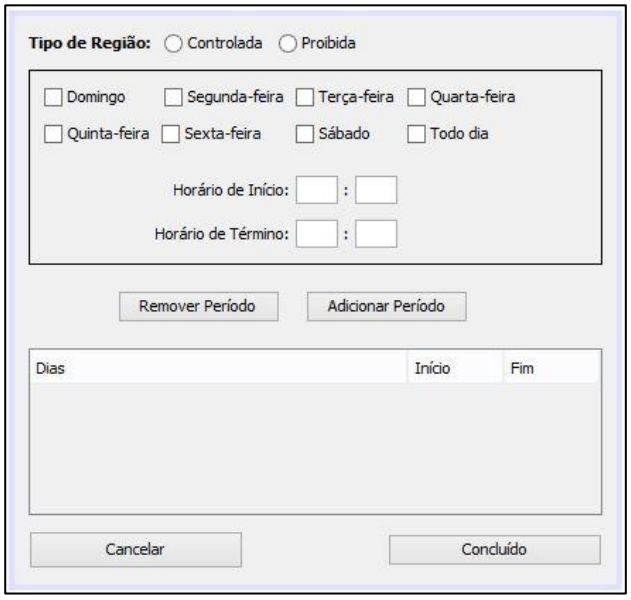

Figura 6. Exemplo de janela para nova região.

Na figura 7 está apresentado um exemplo de criação de uma região para uma câmera.

```
RegionIntruder reg = new RegionIntruder();
Camera cam = reg.loadCamera("IP da câmera");
Region r = new Region();
r.setProhib(true);
Period p = new Period("Dias da semana",horaInicio,
    minutoInicio,horaFim,minutoFim);
ArrayList periodos = new ArrayList();
periodos.add(p);
r.setActive(periodos);
cam.getRegions().add(r);
reg.saveCameras();
```

Figura 7. Exemplo de criação de uma região.

4.1.4 Detecção e Reconhecimento facial

Para a análise da imagem da câmera, é utilizada o método analyzeImage() da classe RegionIntruder. A detecção e o reconhecimento facial em si são realizados pela classe Recognizer. Passando como parâmetro um objeto tipo Camera, este método tem o seguinte funcionamento:

- Primeiramente, é capturada uma imagem atual fornecida pela câmera.
- É realizada a detecção de faces na imagem, utilizando o método detectFaces() da classe Recognizer.
- Para cada região ativa demarcada para esta câmera, são obtidas todas as faces contidas nela.
- São identificadas cada uma dessas faces; caso a região seja proibida, é gerado um log de intrusão para cada uma dessas pessoas; caso a região seja controlada, são gerados os logs

para aqueles que não forem autorizados.

Este processo é realizado de acordo com o diagrama da figura 8.

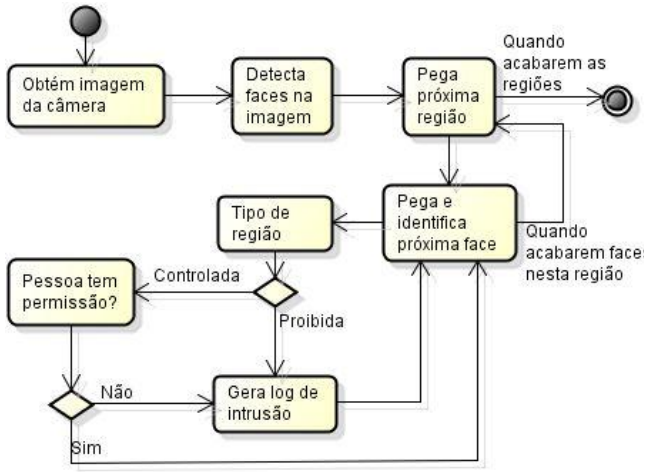

Figura 8. Diagrama de atividade

4.1.4.1 Testes para detecção facial

Para uma melhor precisão da detecção e reconhecimento facial, foram realizados alguns testes sobre algumas imagens.

Na etapa de detecção de faces, o resultado ideal seria que todas e somente as faces contidas na imagem fossem detectadas, ou seja, sem falsos positivos. Para os testes, foram utilizadas 11 imagens obtidas pela câmera principal, contendo quantidades variadas de rostos cada; cada rosto foi classificado como:

- Face fácil: uma face frontal ou levemente inclinada, sem muitas sombras e não muito distante da câmera, que não possui qualidade de imagem muito boa;
- Face difícil: uma face muito inclinada, parcialmente coberta, com sombras em demasia; uma face que até mesmo para seres humanos seria complicado reconhecer rapidamente.

É necessário registrar que cada rosto foi classificado baseado em opiniões humanas; faces que aparentemente são fáceis de ser observadas podem não ser tão triviais para o algoritmo.

Várias combinações de parâmetros foram testadas, e os melhores resultados podem ser observados na tabela 2.

Tabela 2. Resultados dos testes de detecção

Img	Num. Faces	Faces Fáceis	Faces Difíceis	Acertos	Não. Detec.	Erros
1	3	2	1	2	1	0
2	3	2	1	3	0	1
3	4	2	2	3	1	0
4	5	4	1	3	2	1
5	4	3	1	3	1	0
6	5	3	2	3	2	0
7	6	2	4	4	2	0
8	3	2	1	3	0	1
9	3	2	1	3	0	0
10	3	0	3	0	3	2
11	2	2	0	2	0	0

Alguns pontos importantes:

- Imagens idênticas sempre retornam os mesmos resultados, o que indica a consistência do algoritmo.

- Rostos fáceis foram detectados com sucesso na grande maioria das vezes, e uma taxa aceitável de acertos para rostos difíceis.
- O maior problema em todos os testes foi a detecção de falsos positivos em uma grande quantidade de imagens, geralmente nas paredes ou em cartazes. Todas as fotos foram tiradas em horários diferentes mas de uma mesma região, o que indica que a simples mudança na iluminação do sol ou das lâmpadas em algum local da imagem afetava o resultado do algoritmo.
- O classificador utilizado para detectar faces que deu os melhores resultados foi o haarcascade_frontalface_alt.xml, incluso na instalação do OpenCV.

4.1.4.1 Testes para reconhecimento facial

A etapa de reconhecimento facial depende da etapa de detecção facial; como já explicado, a detecção de faces gerou resultados satisfatórios, com boas taxas de acertos em faces fáceis.

A detecção de objetos em imagens utilizando um classificador pronto é simples: regiões das imagens são comparadas com formas pré-definidas e, dependendo da quantidade de similaridades e das distâncias entre essas regiões, o algoritmo considera aquele conjunto de regiões como sendo o objeto sendo procurado. Para o reconhecimento, o processo possui raciocínio semelhante, porém muito mais complexo: a partir de imagens fornecidas para o treinamento, o reconhecedor deve criar uma espécie de classificador interno, que associa uma imagem a uma palavra. Porém, nem sempre as diferenças entre faces de pessoas diferentes são tão acentuadas. Além disso, duas imagens diferentes da face da mesma pessoa podem ser tão diferentes uma da outra, que enganam o algoritmo: o ângulo da foto, a iluminação global, sombras, acessórios como óculos e até mesmo expressões faciais diferentes são alguns fatores que tornam o reconhecimento facial perfeito um sonho distante.

Para atingir uma taxa razoável de acertos no reconhecimento, foram realizados nesse trabalho diversos testes, além da utilização dos resultados de pesquisas de outros autores.

Os testes foram realizados de acordo com os seguintes critérios:

- Para o treinamento, foram utilizadas três pessoas, com duas imagens cada.
- A cada iteração dos testes foram modificados alguns parâmetros: tamanho e resolução das faces, valor limite que indica semelhança entre duas imagens, o raio dos LBP (ver capítulo 3.3.1).

- A cada iteração dos testes também eram modificadas as imagens utilizadas para testar dentre 40 retratos de diversas pessoas.

Apesar dos diversos testes, os resultados encontrados não foram muito diferentes dos resultados iniciais.

Algumas observações sobre os testes no reconhecimento:

- As faces utilizadas no algoritmo, seja para treinamento ou como entrada para identificação devem ser do mesmo tamanho. Esse tamanho não pode ser muito grande, nem muito pequeno; imagens 80 x 80 pixels deram melhores resultados.
- O *threshold*[1] é muito difícil de ser estipulado. Por um lado, ele impede que falsos positivos aconteçam, descartando até a ocorrência mais próxima se esta for muito diferente; por outro lado, o algoritmo descarta imagens muito diferentes de uma mesma pessoa; dependendo da finalidade do programa final, o usuário deve alterar esse valor.
- Com imagens frontais, o algoritmo consegue reconhecer corretamente cerca de 80%. O problema, de novo, são os falsos positivos e negativos, principalmente utilizando fotos de pessoas parecidas.

4.1.5 Intrusão

A intrusão ocorre quando uma pessoa não autorizada ou uma pessoa não identificada aparece em uma região controlada, ou quando qualquer pessoa aparece em uma região proibida.

As informações que um log de intrusão contém podem ser vistas na tabela 1, na classe IntrusionLog.

O usuário tem a opção de desabilitar a geração de logs, caso deseje. O framework salva um XML com as logs a cada cem intrusões. Um exemplo de um desses XML está apresentado na figura 9.

```
<intrusionsLog>
    <intrusion camera="127.0.0.1" type="ctrl">
        <time>31-1-2014-22:45</time>
        <!--opcional-->
        <intruder>Nome do Intruso</intruder>
        <image>ImagemDoIntruso.jpg</image>
        <!---->
    </intrusion>
    <intrusion camera="127.0.0.2" type="prohib">
        <time>4-12-2014-9:25</time>
    </intrusion>
</intrusionsLog>
```

Figura 9. Exemplo XML de intrusão.

4.2 Ferramenta RegionIntruder

Para acompanhar o desenvolvimento do framework, foi também criada uma ferramenta simples, com algumas funcionalidades básicas para demonstrar sua utilização. Foi também através desta ferramenta e outros protótipos que todos os

[1] Valor limite fornecida ao algoritmo que decide se a diferença entre a face treinada mais próxima encontrada e a face de entrada é aceitável

métodos e funcionalidades do framework foram testados e analisados.

A figura 10 apresenta uma imagem da ferramenta exemplo. Existe um painel central, onde as imagens obtidas da câmera são colocadas. Nessa imagem, o usuário pode desenhar as regiões delimitadas para essa câmera, sendo estas apresentadas em uma tabela no canto superior direito.

Alguns botões para as funções básicas do framework estão localizadas à esquerda do painel. Botões para cadastros de novas faces para treinar o algoritmo, para o cadastro de câmeras, para o carregamento de uma câmera, um botão para apenas capturar a imagem atual da câmera, um para salvar manualmente todos os logs gerados e por último um botão para realizar a análise da imagem.

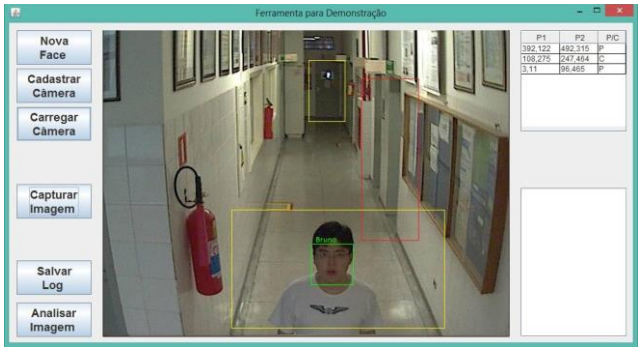

Figura 10. Ferramenta utilizando o framework.

5 CONSIDERAÇÕES FINAIS

O framework final é capaz de realizar funções fundamentais definidas no início do projeto. Algumas funções poderiam ser acrescentadas para melhor clareza: atualmente a mesma função registra uma nova câmera como também altera, sobrescrevendo qualquer arquivo já existente. Outras funções poderiam ser separadas, para que o usuário pudesse realizar apenas parte de um processo, caso queira.

As etapas de detecção e reconhecimento facial foram mais problemáticas. Muitas soluções não atingiam uma taxa de precisão desejável, ou eram muito lentos e mal otimizados. A fase de detecção funciona muito bem para localizar faces, mas ela também encontra alguns falsos positivos. A qualidade das imagens geradas pela câmera é um fator importante, como também a iluminação do local vigiado. O reconhecimento encontra alguns problemas: a foto utilizada para treinar o algoritmo precisa estar semelhante às faces encontradas nas imagens. Isso significa que a qualidade da imagem deve ser semelhante, como também devem ser faces frontais ou com pouca variação no ângulo. Além disso, o algoritmo frequentemente associa rostos detectados a alguma pessoa cadastrada, mesmo que estes rostos sejam desconhecidos; por esse motivo o usuário deve considerar a taxa de confiança na resposta do algoritmo.

Por esses e outros motivos, este trabalho pode ser aperfeiçoado e servir como referência para trabalhos futuros. A imagem obtida da câmera pode passar por processos

de tratamento, como também o algoritmo pode ser alterado por algum outro: muitas soluções existem, sempre com vantagens e desvantagens sobre outras; framework também poderia ser adaptado para funcionar com outros tipos de câmeras, afinal, novas tecnologias e soluções surgem a todo o momento.

REFERÊNCIAS

AMORIM, M.J.V.; BERCHT, M. **O uso da webcam na educação**. Porto Alegre: UFRGS, 2009.

BONSOR, K.; JOHNSON R. **How Facial Recognition Systems Work**. 2011. Disponível em: < http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/facial-recognition.htm>. Acesso em: 08 nov. 2013.

MATURANA, D.; MERY, D.; SOTO, A. **Face Recognition with Local Binary Patterns, Spatial Pyramid Histograms and Naive Bayes Nearest Neighbor classification.** 2009**.** Departamento de Ciencias de la Computación Pontificia Universidad Católica. Santiago, Chile. Disponível em: <http://asoto.ing.puc.cl/papers/Maturana-09.pdf>. Acesso em: 16 nov. 2014

NORMAN, T.L. **Integrated Security Systems Design:** concepts, specifications, and implementation. EUA: Butterworth-Heinemann, 2011.

SILVA, P.Q.; SANTA ROSA, A.N.C. **Reconhecimento facial aplicado à perícia criminal**. In: CONFERÊNCIA INTERNACIONAL DE PERÍCIAS EM CRIMES CIBERNÉTICOS, 1., 2004, Brasília. **Anais...** Brasília: INC/DPF, 2004. Disponível em: <http://www.acmesecurity.org/sites/default/files/publicacoes/artigos/anais-iccyber-dpf-2004.pdf#page=176>. Acesso em: 08 nov. 2013.

SILVA, R. A. **History of CCTV - Closed Circuit Television**. 2009. Disponível em: <http://ezinearticles.com/?History-of-CCTV---Closed-Circuit-Television&id=3118222>. Acesso em: 05 nov. 2013.