Lock-o-tron

Rafael Sá Menezes[1], **Rodrigo dos Santos Tavares**[1], **Herbert Rocha**[1]

[1]Departamento de Ciência da Computação – Centro de Ciência e Tecnologia
Universidade Federal de Roraima (UFRR)
Boa Vista – RR – Brazil

{rafa.sa,rodrigo.tavares, herbert.rocha}@ufrr.br

Abstract. *This paper describes the main features of Lock-o-tron, a security system based on facial recognition, while presenting its algorithms, modules, configuration, connections, libraries used and future works. The algorithms mentioned are Fisherfaces (for facial recognition) and fifo (for real-time scheduling). OpenCV was used as the facial recognition engine, and an Intel Galileo for controlling a motion sensor and an electric lock. At last, the system was able to achieve its goal.*

Resumo. *Este artigo descreve as principais características do Lock-o-tron, um sistema de segurança com reconhecimento facial, abordando seus algoritmos, módulos, disposição, conexões, bibliotecas utilizadas e possibilidades de trabalhos futuros. Os algoritmos descritos são Fisherfaces (para reconhecimento facial) e o de fifo (para escalonamento em tempo-real). Foi utilizado OpenCV para fazer o reconhecimento dos rostos, e um Intel Galileo para controle de um sensor de presença e de uma fechadura elétrica. Ao final, concluímos que fomos capazes de atender a tarefa proposta.*

1. Introdução

O Lock-o-tron é um projeto interdisciplinar que, unindo os conceitos de diversos campos da Computação, realiza o reconhecimento facial de pessoas. O sistema conta com: Um servidor, um Intel Galileo, um Yocto Linux configurado para *soft-realtime*, diversos sensores, diversos atuadores, uma interface Web para administração e um aplicativo Android para controle.

Este trabalho irá descrever as principais características do sistema, assim o leitor poderá entender melhor sobre o funcionamento do sistema como um todo.

2. Dispositivos

O sistema é dividido em duas partes principais: O Galileo, responsável por controlar, e o servidor, responsável por administrar. A figura 1 contém o *Big Picture* do Lock-o-tron.

2.1. Galileo

No Galileo conectamos:

Sensor de Presença: Responsável por detectar se existe alguma pessoa no local para que o sistema saiba quando deve iniciar o reconhecimento facial;

Fechadura elétrica: A fechadura é o atuador final do sistema, sendo ativada para abrir a porta.

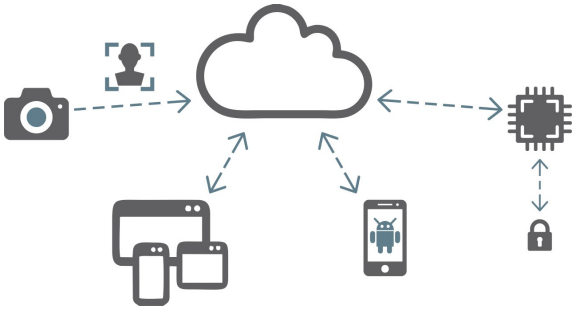

Figura 1. Big Picture

Tabela 1. Orçamento

Produto	**Valor**
Intel Galileo	R$ 499,90
Sensor de presença	R$ 18,90
Câmera	R$ 200,00
Fechadura elétrica	R$ 159,00
Implementaçao do Sistema	R$ 6.120,00
Total:	R$ 6.997,80

2.2. Servidor

No Servidor conectamos:

Câmera: A câmera obtém os frames e os salva no servidor.

2.3. Orçamento

O orçamento foi baseado em que cada programador receberia R$2000,00 mensais (sendo 20 dias úteis), trabalhando 6 horas por dia. O projeto levou em torno de 30 dias para ser desenvolvido. A tabela 1 mostra o orçamento do projeto

3. Comunicação interna

Para comunicação do sistema foram utilizados: um servidor UDP, um *web service* e uma interface Web.

3.1. Servidor UDP

Esse servidor UDP, que é executado no Galileo, é responsável pela interação de outros ambientes com o Galileo. Através destas mensagens UDP é possível abrir a porta e solicitar atualizações. A prioridade do servidor UDP é a mais alta no Galileo, então, sempre que alguém enviar uma mensagem, os outros processos serão interrompidos para que a ação pedida seja executada. A figura 2 mostra a rede de Petri que representa as computações do servidor UDP.

3.2. Web Service

O *web service*, executado no servidor, é responsável pela interação de todos os ambientes entre si. As suas funções são:

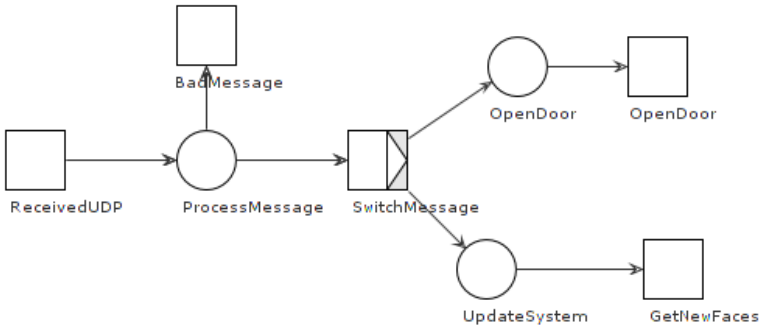

Figura 2. Rede de Petri do servidor UDP

- Enviar frames para o Galileo;
- Enviar atualizações para o Galileo;
- Enviar requisições UDP de abrir porta para o Galileo;
- Enviar requisições UDP de atualizar para o Galileo;
- Alterações no banco de dados;
- Requisições do banco de dados.

3.3. Interface Web

A interface Web, executada no servidor, ficou responsável por ser um meio multiplataforma para modificar o banco de dados.

4. Reconhecimento Facial

Para o reconhecimento facial, é utilizado o algoritmo de *Fisherfaces*. O Galileo obtém o frame do servidor (um frame já pré-processado) e atribui um ID a esse frame. Caso o ID seja 255, o frame é considerado como sendo de ninguém conhecido, caso contrário, é enviada uma requisição ao *web service* para descobrir se a pessoa está autorizada a entrar naquele horário. Caso ela esteja autorizada, a fechadura abre.

Os frames são pré-processados com as seguintes etapas:

1. Aplica-se uma conversão para tons de cinza;
2. Aplica-se uma equalização do histograma no frame;
3. Procura-se os dois olhos da imagem, continua apenas se encontrar;
4. Rotaciona-se o frame para os olhos ficarem alinhados no eixo x;
5. Aplica-se uma equalização do histograma nos lados direito e esquerdo do rosto;
6. Aplica-se o filtro de suavização;
7. Aplica-se uma máscara elíptica para remoção do pescoço.

Tudo isso para normalizar todos os frames e facilitar o reconhecimento facial.

O treinamento é feito ao pegar 60 frames pré-processados (da mesma pessoa) e associa-se essa pessoa a um ID (o mesmo utilizado no banco de dados), então é utilizado o algoritmo de *Fisherfaces* (que já vem implementado no opencv2) para o treinamento e é gerado um xml (treinado) com esses dados. Posteriormente, o Galileo obtém esse XML e aplica o algoritmo de *Fisherfaces* em um frame pré-processado. A figura 3 ilustra a rede de Petri das computações de reconhecimento facial no Galileo.

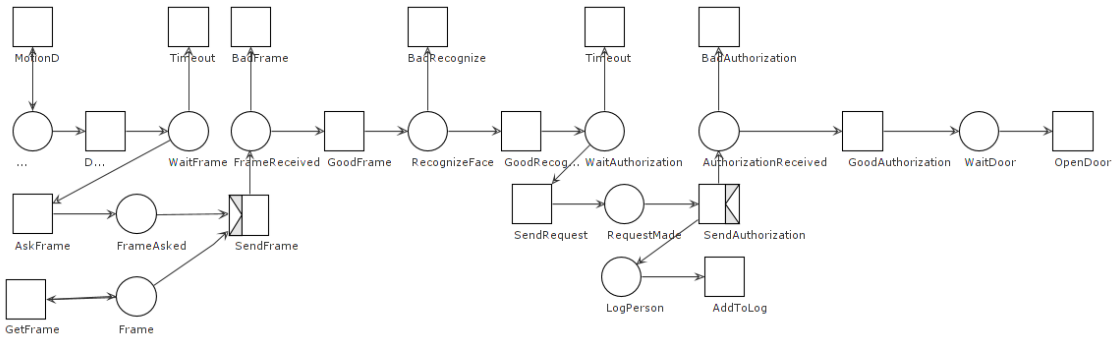

Figura 3. Rede de Petri do Reconhecimento

5. Sistema Operacional

O sistema operacional utilizado no servidor é um GNU/Linux genérico, já no Galileo, é utilizado o Yocto Linux com o kernel configurado para *preempt*.

5.1. Yocto Linux

O Yocto Linux é uma distribuição de Linux desenvolvida especificamente para sistemas embarcados. Para o Galileo, a Intel disponibiliza uma *recipe* específica, que dá suporte à placa e aos seus pinos.

Essa *recipe* adiciona uma biblioteca chamada *mraa*, que facilita a interação das portas, removendo a necessidade de se acessar as GPIO diretamente.

5.2. PREEMPT - Tempo Real

O projeto utiliza um algoritmo de *soft-realtime*, já presente no Linux (configurado com *preempt*), o *fifo*. Este algoritmo tem prioridade sobre qualquer outra forma de escalonamento presente no sistema e executa o escalonamento da seguinte forma:

- Procura-se processo com maior prioridade FIFO (0 - menor, 100 - maior);
- Executa-se o processo até ele terminar ou receber uma mensagem de *yield*;

No Lock-o-tron, foi definido o servidor UDP com uma prioridade de 60 e o reconhecimento facial com prioridade 40. Todos os demais processos do sistema operacional não estão incluídos na tabela de tempo-real, então são os últimos a serem escalonados na tabela.

Por ser um algoritmo de tempo-real estático, o *fifo* não consegue garantir que irá cumprir todos os *deadlines*, mas, os resultados obtidos apenas com ele foram muito positivos.

6. Aplicativo Android

O aplicativo utiliza o *web service* para comunicação com o Galileo e obtenção do histórico. Conta com os seguintes módulos:

Histórico: Menu onde é possível visualizar os últimos usuários que tentaram entrar.
Panic Button: Botão de emergência, quando pressionado a porta abre;
Update Button: Botão para o Galileo atualizar seu arquivo de treinamento;
Estatísticas: Menu onde é possível ver diversas informações sobre o uso do sistema.

7. Trabalhos Futuros

O Lock-o-tron abre portas para diversas possibilidades de expansão:

- Melhorar o algoritmo de reconhecimento;
- Adicionar novos níveis de segurança;
- Verificar número de pessoas que estão entrando em conjunto;
- Um sistema para identificação de padrões de comportamento estranhos.

8. Conclusão

Neste artigo foi apresentado as principais característica do Lock-o-tron, um sistema de segurança com reconhecimento facial, descrevendo seus algoritmos, módulos, disposição e bibliotecas utilizadas.

No final, o sistema criado é capaz de executar a tarefa proposta com certa precisão (ainda sendo ajustada), e o escalonador garante que por mais sobrecarregado o Galileo esteja, as tarefas sempre serão executados em tempo hábil.