[[1]](#footnote-1)

Busca de pessoas em tempo real utilizando openCV em placa Raspberry PI

Kleber de Mattos Dobrowolski¹

***Abstract*—mmmm.**

***Index Terms*—Raspberry, OpenCV, Computational Vision**

*Resumo*—Este documento contém informações de um projeto integrando visão computacional através de biblioteca OpenCV, junto com as características do mundo IoT: hardware de baixo custo (Raspberry PI) e protocolo de comunicação leve (MQtt). Seu diferencial é a utilização do protocolo para envio, em tempo real, de novos modelos de reconhecimento de imagem para o aparelho, assim como a recepção das imagens capturadas, toda a vez que a detecção do alvo for positiva.

1. INTRODUÇÃO

Nos grandes centros urbanos a ocorrência de pessoas perdidas (tais como crianças desgarradas dos pais ou idosos senis que se afastaram do acompanhante) em estabelecimentos de grande área tais como metrô, shopping centers e museus não é incomum (buscar ref). Porém, mesmo com a existência de grande quantidade de equipamentos de vigilância, tais como câmeras, isto não auxilia na busca prática destas pessoas, já que estas câmeras no máximo são utilizadas para busca visual manual por operadores humanos. Para efetuar estas buscas, normalmente são utilizadas características físicas simples, tais como: se o sujeito da busca é uma criança ou adulto; cor da roupa; uso de óculos ou laço no cabelo; características físicas como calvície ou cabelos ruivos.

A abordagem de utilização de hardware de baixo custo para reconhecimento facial já foi apresentada anteriormente por *Chen at al.* [1], portanto o problema de busca pessoas utilizando processo similar parece ser uma tarefa factível, dado que a complexidade do problema é menor.

O projeto proposto é a criação de sistema de reconhecimento de pessoas utilizando abordagem IoT (*Internet of Things*): hardware de baixo custo, comunicação remota em tempo real, aproveitamento de processamento em nuvem. O sistema será composto por nós, contendo cada um uma placa *Raspberry* PI 3, acoplada à sua câmera modular (sensor OV5647, de 5 MP de resolução), utilizando comunicação por rede *WiFi* e protocolo *MQTT* para controlar a atualização dinâmica do algoritmo de busca (modelo). O processamento de imagens é realizado utilizando a biblioteca de domínio público OpenCV (*Open Source Computer Vision Library*).

Convém notar que o foco deste projeto é a integração do *harware* com os conceitos de IoT, sendo a criação dos modelos de reconhecimento de imagem em si, fora do escopo inicial. Porém, observe-se que a criação destes modelos, frequentemente através de técnicas de *machine learning*, poderá ser feita utilizando o poder computacional de processamento em nuvem, podendo o resultado obtido (modelo XML) ser injetado em tempo real nas nos nós, que passarão a buscar o novo sujeito do modelo atualizado, enviando os resultados das buscas para um controlador central (central de controle).



Fig. 1. Resultado ilustrativo de uma busca por criança vestida de cor-de-rosa

1. DESENVOLVIMENTO
   1. *Detalhes da implementação*

A linguagem programação escolhida foi Java, devido à sua boa portabilidade, bem como maturidade e facilidades (como debug no computador desktop e posterior implantação no hardware IoT). Conjuntamente, foram utilizadas as biblioteca PaHo MQtt para comunicação, e biblioteca OpenCV para o processamento de imagens e controle da câmera.

Já o hardware utilizado foi a placa *Raspberry* PI 3, porém note-se que a escolha da linguagem Java permite a utilização deste projeto em dispositivos IoT similares, tais como a placa *SnapDragon* da *Qualcomm*. A câmera acoplada também permite flexibilidade: desde câmeras USB genéricas, até mesmo módulos específicos para hardware IoT, como a Câmera *Raspberry* PI,ou mesmo câmeras com visão noturna.

* 1. *Controle dos nós em tempo real*

Alguns comandos simples foram implementados, sendo os mesmos recebidos através do *topic* do protocolo MQtt.

• ***exit*** e ***restart***: desligam e reiniciam, respectivamente, o programa Java.

• ***read*** e ***write***: permitem ler e escrever parâmetros do arquivo de configuração (chamado argos.properties). Exemplos destes parâmetros: endereço IP do *broker* MQtt, iD individual do nó, nome da classe Java OpenCV utilizada na detecção de imagens. Alguns destes comandos demandam reinício do programa (tal como mudança do endereço IP do broker), através do comando *restart*, acima. O novo conteúdo destes parâmetros é recebido dentro da parte *payload* do protocolo MQtt.

• ***snapshot***: força o nó a enviar uma foto atual, com upload através de protocolo MQtt, dentro do campo de *payload* (observação: pode existir limitação de tamanho da mensagem, dependendo do *broker* utilizado).

• ***class***: comando para receber um novo arquivo OpenCV. Este arquivo será imediatamente compilado dentro da placa *Raspberry* PI, e carregado através da técnica de Java *Reflexion*.

* 1. *Escopo dos comandos e GUI*

A utilização do protocolo MQtt permite grande flexibilidade no envio de comandos: podemos definir escopos como nós individuais, grupos de nós, ou todos os nós. Para enviar comandos para um nó específico podemos usar o Id do mesmo (/cam\_id37/command/restart). Também é possível enviar comandos para todas os nós acoplados ao sistema (/command/snapshot), bastando para isto escolher o tópico MQtt mais genérico.

Uma interface GUI rudimentar foi criada, apenas para possibilitar a visualização de fotos de maneira rápida. Um sistema profissional necessitaria melhorias consideráveis nesta GUI.

* 1. *Cenário De Operação*

O projeto é flexível o bastante para atender muitos cenários operacionais (monitoração de metrô, sistemas distribuídos em ônibus, etc). Para um exemplo mais concreto, descreveremos a seguir uma operação em um shopping center hipotético, onde:

• Diversos módulos estão distribuídos pelos departamentos do shopping, sejam acoplados a câmeras de segurança já existentes ou substituindo as mesmas. Comunicam-se por MQtt, usando a rede TCP/IP preexistente do shopping. No Centro de Controle (CO) existe um *harware* mais poderoso, para o controle centralizado dos módulos e, opcionalmente, com ferramentas mais poderosas de *Machine Learning*, para criar novos perfis baseados em fotos, em poucos minutos.

• o CO recebe uma demanda: uma criança de aproximadamente 4 anos, vestida de vermelho, perdeu-se da mãe. Uma foto da criança é repassada ao CO, através de do celular de um dos seguranças do shopping. CO decide qual a melhor abordagem: criar um modelo ‘haarcascade’ novo (consistindo de uma classe Java + arquivo xml) personalizado com a foto da criança, ou apenas utilizar um dos modelos padrão existentes, porém ajustado (‘criança’ + ‘roupa vermelha’).

• Imediatamente, os nós recebem a nova classe Java através de MQtt, compilam e carregam na memória, passando a executá-la. As câmeras que detectarem crianças que se enquadrem nas novas características, começarão a enviar imagens periodicamente (arquivos no formato .JPG) com a localização do setor e a hora, para o CO.

• Como, provavelmente, existirão algumas detecções falso-positivo, um operador do CO fará a inspeção visual final, para confirmar se a criança foi realmente encontrada. Se sim, enviará a foto da detecção para o segurança mais próximo do setor, que poderá abordar a criança e trazê-la com segurança ao CO (ou acompanha-la diretamente ao encontro de sua mãe).

1. Desenvolvimentos futuros

Como sugestão de futura continuação deste projeto, temos a criação de uma interface de usuário mais amigável e integradas com ferramentas de nuvem, para a geração das classes de busca.

.

1. Referências
2. Chen, Yong-Ping at al. “Low-Cost Face Recognition System Based on Extended Local Binary Pattern”, in International Conference on Automatic Control Conference, p. 13–18, novembro. 2016.
3. Nguyen. Huu-Quoc at al. Low Cost Real-Time System Monitoring Using Raspberry Pi. Em 2015 Seventh International Conference on Ubiquitous and Future Networks, 7-10 July 2015Aaba

¹ **Kleber de Mattos Dobrowolski** nasceu em Taubaté, SP, em 03 de novembro de 1975. Possui os títulos: Bacharel em Computação Científica (UNITAU, 2000) e Mestrado em Computação Aplicada (INPE, 2004).

De 2003 a 2006 foi analista de sistemas no INPE, na Divisão de Desenvolvimento de Sistemas Solo - DSS. Trabalha desde 2006 no Ericsson Brasil, em desenvolvimento de software para bilhetagem de celulares. Tem interesse nas áreas de Redes Neurais, Computação Aplicada e Internet das Coisas

1. The ESP8266 funciona com 802.11n. [↑](#footnote-ref-1)